B.Tech – COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY) PROGRAMME

CBCS CURRICULUM

Specialization / Honors

(With effect from 2022-2023)

Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

After the completion of the degree, graduates will

PEO1: Formulate, solve and analyze Computer Science and Engineering problems using necessary mathematical, Scientific and engineering fundamentals.

PEO2: Equip with fundamental knowledge to solve problems in the domain of Cyber security and Cyber forensics.

PEO3: Excel as software engineer in the specialized field of cyber security or continues higher education at a reputed institution in India or abroad.

PEO4: Demonstrate critical thinking, communication, teamwork, leader ship skills and ethical behaviour necessary to function productively and professionally.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

PO1. Engineering knowledge: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.

PO2. Problem analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

PO3. Design/development of solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)

PO4. Conduct investigations of complex problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8)

PO5. Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6. The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7)

PO7. Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8. Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9. Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences.

PO10. Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11. Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change. (WK8)

PROGRAM SPECIFIC OUTCOMES (PSO)

On successful completion of the program, the graduates will be able to,

PSO1: Develop the applications in various domains of computer science and engineering using appropriate algorithms and techniques.

PSO2: Use tools and techniques to provide secure solutions for offensive and defensive cyber security.

COURSE OUTCOMES (COs)

Abilities of the student defined in terms of Course Outcomes (COs) as per the Bloom's Taxanomy at the end of every course in the programme.

B.Tech –Computer Science and Engineering (Cyber Security)
Curriculum
(CBCS)
Honors / Specialization
(With effect from 2022-2023)

Credits required for regular students in various course categories for B.TechComputer Science and Engineering (Cyber Security)

Preamble:

B.Tech. CSE (Cyber Security) is designed to address the industry's increasing demand for skilled security professionals in the public and private sector, both in the Data Security and in the Network/Cloud Security domains. The programme covers core computer science subjects as well as Cyber Security specific courses. The emphasis of the program is to nurture students with the knowledge and skills required to secure computers, detect and analyze attacks and threats, respond to attacks, develop security policies, procedures, and standards.

The students shall earn 164 credits in various course categories given below for the award of degree of B.Tech (Computer Science and Engineering (Cyber Security)).

Course Category	Minimum Credits Required
Foundation Courses (FC)	56
Program Core (PC)	58
Program Elective (PE)	18
Open Elective (OE)	12
Independent Learning (IL)	14
Industry / Higher Institute Learning Interaction (IHL)	2
Professional Proficiency Courses (PPC)	4
TOTAL	**164**

Minimum credits required for regular students in various course categories for B.Tech Computer Science and Engineering (Cyber Security) with minor

The students shall earn 164 credits in various course categories and additional 18 to 20 credits in the specialized tracks / areas from other branches/Schools by satisfying the prerequisite courses for the award of degree of B.Tech Computer Science and Engineering (Cyber Security) with minor subject to the regulations.

Minimum credits required for regular students in various course categories for B.Tech Computer Science and Engineering (Cyber Security) with Honors

The students shall earn 164 credits in various course categories and additional 18 to 20 credits in the specialized tracks / areas courses by satisfying the prerequisite courses for the award of degree of B.Tech Computer Science and Engineering (Cyber Security)with Honors subject to the regulations.

Foundation Core (56 Credits)

Foundation courses enhance the knowledge, skills and attitude of UG engineering graduates of all programmes to the expected level. The foundation courses shall have the courses related to basic sciences and mathematics, basic engineering sciences and humanities and social sciences.

L-Lecture, T-Tutorial, P-Practical, C-Credit

S.No	Course Code	Subject Title	Category	L	T	P	C
Lecture Courses							
1	10210MA101	Linear Algebra for Computing	BSC	3	1	0	4
2	10210MA102	Calculus & Ordinary differential Equations	BSC	3	1	0	4
3	10210MA103	Probability, Statistics and Queuing theory	BSC	3	1	0	4
4	10210MA110	Discrete Mathematical Structures	BSC	3	1	0	4
5	10210PH101	Semiconductor Physics	BSC	3	0	0	3
6	10210CH104	Environmental Science and Sustainability	BSC	3	0	0	3
7	10210CS101	Problem Solving using C	ESC	3	0	0	3
8	10210CS104	Programming using Python	ESC	3	0	0	3
9	10210ME101	Design thinking	ESC	2	0	0	2
10	10210BM101	Biology for Engineers	ESC	2	0	0	2
11	10210ME103	Innovation & Entrepreneurship	ESC	2	0	0	2
12	10210ME102	Universal Human Values	HSC	3	0	0	3
13	10210ME104	Project Management and Finance	HSC	2	0	0	2
14	10210ME105	Engineers and Society	HSC	1	0	0	M
15	10210BL101	Constitution of India	HSC	1	0	0	M
Integrated Courses							
16	10210EN201	Professional Communication - I	HSC	1	0	2	2
17	10210EN202	Professional Communication - II	HSC	1	0	2	2
18	10210EC201	Basic Electronics & Digital Logic Design	ESC	2	0	2	3
19	10210EE204	Introduction to Engineering	ESC	1	0	4	3
20	10210ME201	Engineering Graphics	ESC	1	0	4	3
Laboratory Courses							
21	10210PH301	Modern Physics Laboratory	BSC	0	0	2	1
22	10210EE301	Engineering Products Lab	ESC	0	0	2	1

23	10210CS301	Problem Solving using C Lab	ESC	0	0	2	1
24	10210CS305	Programming using Python Laboratory	ESC	0	0	2	1
Total Credits							**56**

*****BSC – Basic Science Courses, ESC – Engineering Science Courses, HSC – Humanities & Social Science Courses, M – Mandatory course

Program Core (58 Credits)

L-Lecture, T-Tutorial, P-Practical, C-Credit

S.No	Course Code	Course Name	L	T	P	C	PG NO
		Theory Courses					
1	10211CC101	Data Structures	3	0	0	3	15
2	10211CC103	Operating Systems	3	0	0	3	18
3	10211CC129	Modern Computer Architecture	3	0	0	3	20
4	10211CC130	Fundamentals of Computer Networks	3	0	0	3	23
5	10211CC106	Formal Languages and Automata Theory	3	0	0	3	26
6	10211CC107	Compiler Design	3	0	0	3	29
7	10211CC109	Microprocessors	2	0	0	2	32
8	10211CC119	Cryptography and Network Security	3	0	0	3	35
		Integrated Courses					
9	10211CC202	Design and Analysis of Algorithms	3	0	2	4	38
10	10211CC204	Programming Using Java	3	0	2	4	45
11	10211CC207	Database Management Systems	3	0	2	4	49
12	10211CC208	Software Engineering	2	0	2	3	55
13	10211CC210	Big Data Analytics	3	0	2	4	59
14	10211CC212	Web and Mobile Application Development	3	0	2	4	64
15	10211CC219	Cyber Security	3	0	2	4	71
16	10211CC224	Full Stack Application Development	1	0	4	3	72
17	10211CC225	Problem Solving and Testing using Java	1	0	4	3	80
18	10211CC226	Java Programming	3	0	2	4	95
		Laboratory Courses					
16	10211CC301	Data Structures Laboratory	0	0	2	1	108
17	10211CC312	Fundamentals of Computer Networks Laboratory	0	0	2	1	113
18	10211CC304	Operating Systems Laboratory	0	0	2	1	119
19	10211CC305	Microprocessors Laboratory	0	0	2	1	125
20	10211CC306	Competitive Coding-I	0	0	2	1	128
21	10211CC307	Competitive Coding-II	0	0	2	1	132
22	10211CC310	IoT and Cloud Laboratory	0	0	2	1	139
23	10211CC313	Problem Solving Techniques	0	0	2	1	142
		Total Credits				**58**	

Tutorial hour is not considered for credit calculation of the course

B.Tech –Computer Science and Engineering (Cyber Security)

CBCS CURRICULUM

Honors / Specialization

Minimum credits required for Lateral Entry students in various course categories for B.Tech CSE(Cyber Security)- VTR UGE 2021

The students shall earn 120 credits in various course categories given below for the award of degree of B.Tech CSE(Cyber Security).

Course Category	Minimum Credits Required
Foundation Courses (FC)	22
Program Core (PC)	48
Program Elective (PE)	18
Open Elective (OE)	12
Independent Learning(IL)	14
Industry / Higher Institute Learning Interaction(IHL)	2
Professional Proficiency Courses (PPC)	4
TOTAL	**120**

B.Tech –Computer Science and Engineering (Cyber Security)

CBCS CURRICULUM

Honors / Specialization

VTR UGE 2021

Foundation Core Course : 22 Credits

L-Lecture T-Tutorial P-Practical C-Credits

S.No	Course Code	Subject Title	Category	L	T	P	C
Lecture Courses							
1	10210MA110	Discrete Mathematical Structures	BSC	3	1	0	4
2	10210CH104	Environmental Science and Sustainability	BSC	3	0	0	3
3	10210CS104	Programming Using Python	ESC	3	0	0	3
4	10210ME101	Design thinking	ESC	2	0	0	2
5	10210BM101	Biology for Engineers	ESC	2	0	0	2
6	10210ME103	Innovation & Entrepreneurship	ESC	2	0	0	2
7	10210ME102	Universal Human Values	HSC	3	0	0	3
8	10210ME104	Project Management and Finance	HSC	2	0	0	2
9	10210ME105	Engineers and Society	HSC	1	0	0	M
10	10210BL101	Constitution of India	HSC	1	0	0	M
Laboratory Courses							
11	10210CS305	Programming Using Python Lab	ESC	0	0	2	1
Total Credits							**22**

Program Core – 48 Credits

L-Lecture, T-Tutorial, P-Practical, C-Credit

S.No	Course Code	Course Name	L	T	P	C	PG NO
		Theory Courses					
1	10211CC101	Data Structures	3	0	0	3	15
2	10211CC129	Modern Computer Architecture	3	0	0	3	20
3	10211CC130	Fundamentals of Computer Networks	3	0	0	3	23
4	10211CC106	Formal Languages and Automata Theory	3	0	0	3	26
5	10211CC107	Compiler Design	3	0	0	3	29
6	10211CC119	Cryptography and Network Security	3	0	0	3	35
		Integrated Courses					
9	10211CC202	Design and Analysis of Algorithms	3	0	2	4	38
10	10211CC204	Programming Using Java	3	0	2	4	45
11	10211CC207	Database Management Systems	3	0	2	4	49
12	10211CC208	Software Engineering	2	0	2	3	55
13	10211CC210	Big Data Analytics	3	0	2	4	59
14	10211CC212	Web and Mobile Application Development	3	0	2	4	64
15	10211CC219	Cyber Security	3	0	2	4	71
16	10211CC224	Full Stack Application Development	1	0	4	3	72
17	10211CC225	Problem Solving and Testing using Java	1	0	4	3	80
18	10211CC226	Java Programming	3	0	2	4	95
		Laboratory Courses					
16	10211CC301	Data Structures Laboratory	0	0	2	1	108
17	10211CC312	Fundamentals of Computer Networks Laboratory	0	0	2	1	113
18	10211CC310	IoT and Cloud Laboratory	0	0	2	1	139
		Total Credits				**48**	

Tutorial hour is not considered for credit calculation of the course

Program Electives (18 Credits)

ProgramElectives are the courses offered in the programme which covers depth and breadth. The students may register for appropriate electives offered in the programme based on their area of interest. One course under this category shall be taken from the list of approved MOOCs.

L-Lecture, T-Tutorial, P-Practical, C-Credit

S.No	Course Code	Course Name	L	T	P	C	PG NO
		Cyber Security – Core Electives					
1	10212CC112	Cyber Security Policy, Law & Ethics	3	0	0	3	
2	10212CC126	Information Security	3	0	0	3	
3	10212CC123	Data Privacy and Security	2	0	0	2	146
4	10212CC211	Artificial Intelligence Techniques	3	0	2	4	149
5	10212CC225	Ethical Hacking	3	0	2	4	154
6	10212CC226	Secure Coding	3	0	2	4	158
7	10212CC228	Blockchain Technology	2	0	2	3	161
8	10212CC230	Forensics in Cyber Security	3	0	2	4	165
9	10212CC232	Vulnerability Management and Penetration Testing	3	0	2	4	169
10	10212CC250	Identity Access Management	3	0	2	4	
11	10212CC292	Applied Programming Skills	3	0	2	4	173

Open Electives (12 Credits)

Open electives are the courses offered across the schools to enhance the knowledge breadth and professional competency of the students. The students shall register for appropriate electives offered in other schools based on their area of interest. The courses offered under this category cover the interdisciplinary knowledge.

L-Lecture, T-Tutorial, P-Practical, C-Credit

S.No	Course Code	Course Name	L	T	P	C
1	XXX1	Course Name-1	3	0	0	3
2	XXX2	Course Name-1	3	0	0	3
3	XXX3	Course Name-1	3	0	0	3
4	XXX4	Course Name-1	3	0	0	3

* One of the courses may be completed through MOOCs Platform like NPTEL as described by the department.

These courses offered to the other departments/schools by School of Computing under Open Elective category.

L-Lecture, T-Tutorial, P-Practical, C-Credit

S.No	Course Code	Course Name	L	T	P	C
1	10213CC101	Object Oriented Programming using Java	3	0	0	3
2	10213CC102	Data Structures	3	0	0	3
3	10213CC103	Operating Systems	3	0	0	3
4	10213CC104	Database Management Systems	3	0	0	3
5	10213CC105	Computer Networks	3	0	0	3
6	10213CC106	Data warehousing and Data mining	3	0	0	3

Independent Learning (14 Credits)

The students are expected to learn the courses offered under this category on their own. The courses offered under this category include:

L-Lecture, T-Tutorial, P-Practical, C-Credit

S.No	Course Code	Course Name	L	T	P	C
1	10214CC501	Community Service Project	0	0	2	1
2	10214CC601	Minor Project-I	0	0	4	2
3	10214CC602	Minor Project-II	0	0	4	2
4	10214CC701	Major Project	0	0	18	9

Industry / Higher Institute Learning Interaction (2 Credits)

The students shall earn a minimum of two credits by undergoing internship and/or specialized courses.

1. **Internship:**
   The students shall undergo Internship in the industry/higher learning institute approved by Industry-Institute Interaction Cell (IIIC) during any time after the second academic year.

2. **Specialized Courses:**
   The students shall undergo the courses offered either by the industrial experts whose minimum academic qualification is Bachelor of Engineering or equivalent or faculty expert from higher learning institutions approved by IIIC. The students shall choose either one two credits course or one one credit course or two one credit courses.

L-Lecture, T-Tutorial, P-Practical, C-Credit

S.No	Course Code	Course Name	L	T	P	C
1	10215CC901	Internship	-	-	-	2
2	10215CC902	Industry Expert Lecture-1	-	-	-	1
3	10215CC903	Industry Expert Lecture-2	-	-	-	1
4	10215CC951	Higher Institute Learning Interaction-1	-	-	-	1
5	10215CC952	Higher Institute Learning Interaction-2	-	-	-	1

Professional Proficiency Courses (4 Credits)

The Professional Proficiency Courses which carry four credits, to be offered in four different semesters, starting from third semester. These courses offered in this category are relevant to professional proficiency.

L-Lecture, T-Tutorial, P-Practical, C-Credit

S.No	Course Code	Course Name	L	T	P	C
1	10216GE901	Soft Skill-I	2	0	0	1
2	10216GE902	Soft Skill-II	2	0	0	1
3		Professional Proficiency Course-III	2	0	0	1
4		Professional Proficiency Course-IV	2	0	0	1

PROGRAM CORE

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC101	Data Structures	3	0	0	3

A.Preamble

This course aims at moulding the learner to understand the various data structures, their organization and operations. The course helps the learners to assess the applicability of different data structures and associated algorithms for solving real world problem.

B.Prerequisite Course

10210CS101 - Problem Solving using C

C.Course Objectives

Learners are exposed to

- Understand basic elementary data structures.
- Illustrate the basic properties of Trees and Graphs.
- Implement various techniques of sorting and searching algorithms.
- Choose efficient data structure and apply them to solve problems.

D.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No’s	Course Outcomes	K - Level
CO1	Design modular algorithms to find solution for computational problem with time and space complexities using suitable linear data structure.	K3
CO2	Demonstrate the familiarity with tree data structure, rule to manipulate those, and their canonical applications.	K3
CO3	Solve unstructured problem using balanced search tree algorithm, hashing function.	K3
CO4	Implement an appropriate algorithm using graph ADT for an application.	K3
CO5	Apply sorting and searching techniques for real world problems.	K3
Knowledge Level (Based on revised Bloom’s Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of Cos with Program Outcomes and Programme Specific Outcomes

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	3	2							3	3	2
CO2	3	3	3	2							3	3	2
CO3	3	3	3	3							2	3	2
CO4	3	3	3	3							3	3	2
CO5	3	3	2	2							2	3	3

High - 3, Medium - 2, Low - 1

F.Course Contents

Unit 1 Introduction to Data Structure L–9 Hours

Introduction: Dynamic aspects of operations on data, Characteristics of data, Creation, Manipulation and Operations on data, Data Structure and its types, Abstract Data Types (ADTs), Analysis of algorithms and its Types, Asymptotic Notation. Arrays: Allocation, Operations and Storage with one-dimensional arrays and multidimensional arrays. Stacks: Operations on Stacks, Applications of Stacks, Queues: Operations and its types. Linked lists: Operations and Types of linked list (singly, doubly and circularly linked list), Implementation of Stack and Queue using linked list, Applications of Linked List.
Case Study: Tower of Hanoi, Sparse matrix.

Unit 2 Trees L–9 Hours

Introduction to Trees – Definitions and concepts, Representation of Binary Tree, Types of Binary tree, properties, structure and applications of binary tree, Binary Tree Traversal with Recursive and Non-Recursive - Expression Tree, Binary Search Trees – Definition, Properties and Operations of binary search tree. Priority Queue: Operations, Implementations- Heaps: properties and types of Heaps, Binary Heap: Representation, Declaration and Creation of Heap, Heapifying, Inserting and Deleting an element, Destroying Heap, Heap Sort. Case Study: Priority Queue in bandwidth management

Unit 3 Special Trees & Hashing L–9 Hours

AVL Tree: Properties, Declarations, Rotations, Insertion, Deletion. B-Tree: Properties, Insertion, Deletion –Trie: Insertion, Deletion, Searching. Hashing - Separate Chaining – Open Addressing – Linear Probing – Quadratic Probing – Double Hashing –Rehashing. Case Study: DNA sequence matching

Unit 4 Graphs L–9 Hours

Introduction to Graphs – Operations on graphs – Representation of Graph – Topological Sort - Graph Traversal - Depth first search – Breadth First Search – Minimum Spanning Tree – Prim's Algorithm- Kruskal's Algorithm – Disjoint subsets and Union-Find algorithms – Shortest Path Algorithms – Unweighted Shortest Paths –Dijkstra's Algorithm. Applications of Graphs. Case Study: Representing Google maps, Recommendations on e-commerce websites.

Unit 5 Sorting, Searching L–9 Hours

Searching Techniques: Linear Search, Binary Search. Sorting techniques: Insertion Sort, Selection sort, Merge sort, Quick Sort, Shell sort, Bubble sort, Counting sort, Bucket sort, Radix sort. Case Study: Tim Sort

Total: 45 Hours (L)

F. Learning Resources

i. Text Books:

1. M. A. Weiss, "Data Structures and Algorithm Analysis in C", Second Edition, Pearson Education, Reprinted edition 2013. [Unit 1,2,3]
2. Ellis Horowitz, Sartaj Sahni, Susan Anderson Freed, "Fundamentals of DataStructure in C", Universities Press,2017. [Unit 4,5]

ii. Reference Books:

1. V. Aho, J. E. Hopcroft, and J. D. Ullman, "Data Structures and Algorithms",Pearson Education, First Edition, Reprint 2017. [Unit 1,2]
2. R. F. Gilberg, B. A. Forouzan, "Data Structures", Second Edition, Thomson India Edition, 2007. [Unit 3,4]
3. Seymour Lipschutz, "Data Structures with C", Scham's outlines, Tata McGraw Hill, 2017. [Unit 4,5]

iii. Online References:

1. "Introduction to Data Structures and Algorithms" 31.12.2009. Accessed on April 2020, [Online]. Available: https://nptel.ac.in/courses/106/102/106102064/.
2. "Data Structures and algorithms". Accessed on: April 15, 2021 [Online]. Available: https://www.coursera.org/specializations/data-structures-algorithms.
3. "Data structures Visualization" 2011. Accessed on: April 15, 2021 [Online]. Available: https://www.cs.usfca.edu/~galles/visualization/Algorithms.html
4. "Data Structures Fundamentals" 2021, Accessed on: April 20,2021 [Online]. Available: https://www.edx.org/course/data-structures-fundamentals

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC103	Operating Systems	3	0	0	3

A.Preamble

This course provides discussion about Operating systems kernel architecture, address spaces, system call interface, processes & threads, inter process communication, CPU scheduling deadlock, scheduling, main memory, virtual memory and file systems. This course aims to understand how Operating system makes computer system convenient to use in an efficient manner.

B.Prerequisite Course

10211CC101 - Data Structures

C.Course Objectives

Learners are exposed to

- Recognize operating Systems basic concepts, structures and System Calls.
- Apply process management concepts and Synchronization techniques.
- Learn different CPU scheduling algorithms and Deadlock Handling methods.
- Understand Various Memory and File management techniques.

D.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No’s	Course Outcomes	K - Level
CO1	Describe the operating system structures,operations and system calls.	K2
CO2	Demonstrate process management concepts and process synchronization methods for real time problems.	K3
CO3	Illustrate CPU scheduling algorithms and deadlock handling methods for the given situation.	K2
CO4	Explain the concepts of various memory management techniques.	K2
CO5	Discussthe concepts of disk management and file system interface.	K2
Knowledge Level (Based on revised Bloom’s Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of COs with Program outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3											2	
CO2	2	3	2	1								2	
CO3	1	3	3	1								2	
CO4	2	3	3									3	
CO5	3	3	3									2	

High- 3; Medium-2; Low-1

F.Course Contents

Unit 1 Structure and Overview of Operating Systems L-8 Hours

Operating system overview: Objectives – functions - Computer System Organization - operating System Operations- System Calls, System Programs-Operating- System Structure: Traditional UNIX system structure-The Mac OS X structure - Architecture of Google's Android.

Unit 2 Process Management L-10 Hours

Processes: Process Concept – Threads - Process Scheduling - Operations on Processes – Inter process Communication - Communication in Client–Server Systems-Pipes (RPC, Pipes) - Process Synchronization: The Critical-Section Problem - Semaphores – Mutex Locks- Classic Problems of Synchronization – Monitors. Case Study: Windows Threads and Linux Threads

Unit 3 CPU Scheduling and Deadlock Management L-9 Hours

CPU Scheduling: Scheduling Criteria - Scheduling Algorithms. Deadlocks: Deadlock Characterization - Methods for Handling Deadlocks - Deadlock Prevention - Deadlock Avoidance - Deadlock Detection - Recovery from Deadlock. Case Study: Real Time CPU scheduling

Unit 4 Memory Management L-9 Hours

Main Memory: Swapping - Contiguous Memory Allocation, Segmentation, Paging - Structure of the Page Table - Virtual Memory: Demand Paging - Page Replacement - Allocation of Frames – Thrashing. Case study: Virtual machine

Unit 5 Storage Structure & File Systems L-9 Hours

Mass Storage Structure: Disk Structure - Disk Scheduling - Disk Management-Structure - File-System Interface: File Concepts -Directory Structure - File Sharing – Protection- File Allocation Methods-NFS. Case study: Recovery in Windows.

TOTAL: 45 Hours

G.Learning Resources

i.Text Book:

1. Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, "Operating System Concepts", 9th Edition, John Wiley and Sons Inc., 2016. [Unit 1-5]

ii.Reference Books:

1. Andrew S. Tanenbaum, "Modern Operating Systems", 4th Edition, Prentice Hall, Wesley, 2014.
2. Thomas Anderson and Michael Dahlin,"Operating Systems: Principles and Practice", 2nd Edition.

iii.Online References:

1. "Learn Operating System" Accessed on: April 21, 2021 [Online]. Available:https://www.tutorialspoint.com/operating_system/index.htm
2. Prof.Santanu Chattopadhyay, "Operating System Fundamentals" July 25,2019. Accessed on: April 21, 2021 [Online]. Available:https://nptel.ac.in/courses/106/105/106105214/

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC129	Modern Computer Architecture	3	0	0	3

A.Preamble

This course specially focuses on the basic structure and functional units of a digital computer. It will explore the interaction of hardware and software and consider the efficient use of hardware to achieve high performance. Moreover, this course describes the parallel organisation to enhance the performance. The topic of memory technologies, memory hierarchy; multi-core processors, GPU, and modern technological advancements are presented.

B.Prerequisite Course

10210EC201- Basic Electronics & Digital Logic Design

C.Course Objectives

Students are able to

- Understand basic principles of Computer Systems.
- Explain various logic design techniques and their applications.
- Use high performance computing architecture.
- Implement CUDA Programming model for Parallel computing

D.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Understand the organization of the Control unit, Arithmetic and Logical unit, Memory unit and the I/O unit.	**K2**
CO2	Analyse different computer architectures and applications.	**K3**
CO3	Explain modern design structures of Pipelined and Multiprocessors systems.	**K2**
CO4	Know distributed computing architecture and high-performance computing.	**K2**
CO5	Implement the CUDA Programming model for Parallel computing.	**K2**
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of COs with Program outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	3			1							
CO2	3	3	2			1							
CO3	3	3				1						1	
CO4	3	3				1						3	
CO5	3	3	3	3	1	1						3	

High- 3; Medium-2; Low-1

F.Course Contents

Unit 1 Basic Structure of a Computer System 9 Hours

Functional Units, Basic Operational Concepts, Instructions, Instruction’s cycle, Memory reference instruction, I/O interruption, Adder and Subtractor circuits, Booth Multiplication Algorithm, Pipelining Review, control hazards and the motivation for caches, cache characteristics and basic superscalar architecture basics.

Unit 2 Multi Core Architecture 9 Hours

Memory technologies, hierarchical memory systems, the locality principle and caching, direct- mapped caches, block size, cache conflicts, associative caches, write strategies, advanced optimisations, performance improvement techniques, DRAM – organization, access techniques, scheduling algorithms and signal systems. Tiled Chip Multicore Processors (TCMP), Network on Chips (NoC), NoC router – architecture, design, routing algorithms and flow control techniques, Advanced topics in NoC and storage – compression, prefetching, QoS.

Unit 3 Distributed Computing Systems and Concurrency 9 Hours

Relation to Parallel Multiprocessors/multicomputer Systems, Distributed and Concurrent Programs, Message Passing vs. Shared Memory Systems, Synchronous vs. Asynchronous Executions, Design Issues and Challenges, Distributed Computing Technologies, Clocks and Synchronization, Coordination and Agreement Algorithms, Global State and Distributed Transactions.

Unit 4 High Performance Computing (HPC) 9 Hours

HPC Architecture, Parallel Processing, Parallel Memory Models, Data vs. Task Parallelism, High Throughput Computing, Vectorization, Multithreading.

Unit 5 High Performance Computing with CUDA 9 Hours

CUDA programming model, Basic principles of CUDA programming, Concepts of threads and blocks, GPU and CPU data exchange.

Total: 45 Hours

G. Learning Resources

i. Text Books:

1. Hamacher, V.C., Vranesic, Z.G., and Zaky, S.G., "Computer Organization", 5/e. McGraw-Hill. 2014. [Unit- 1,2,5]
2. Patterson, D.A., and Hennessy, J.L. , "Computer Architecture : A Quantitative Approach ", Morgan Kaufmann Publishers, , Inc.2017.[Unit -3,4]

ii. Reference Books:

1. Patterson, D.A., and Hennessy, J.L."Computer Organization and Design: The Hardware/Software Interface", Morgan Kaufman Publishers, 5th Edition, Inc.2013
2. Stalling W, "Computer Organization and Architecture ", Pearson Education India. 10th Edition, 2016

iii. Online References:

1. Computer Architecture and Organisation" June, 8, 2017. Accessed on April, 20, 2021 [Online]. Available: https://nptel.ac.in/courses/106/105/106105163/
2. "Introduction to Computer Architecture " January, 1, 2017. Accessed on June, 30, 2021 [Online]. Available: https://nptel.ac.in/courses/106/102/106102157/
3. "GPU Architectures and Programming" April, 17, 2020. Accessed on December, 14, 2022 Available: https://onlinecourses.nptel.ac.in/noc20_cs41/preview
4. https://onlinecourses.nptel.ac.in/noc20_cs41/preview
5. https://www.coursera.org/learn/introduction-high-performance-computing#syllabus

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC130	Fundamentals of Computer Networks	3	0	0	3

A. Preamble

This course is to provide students with an overview of the concepts and fundamentals of computer networks. Topics to be covered include: data communication concepts and techniques in a layered network architecture, communications switching and routing, types of communication, network congestion, network topologies, network configuration and management, network model components, layered network models (OSI reference model, TCP/IP networking architecture) and their protocols, various types of networks and their protocols.

B. Prerequisite Courses

10210CS101-Problem Solving Using C

C. Course Objectives

Learners are exposed to

- Build an understanding of the fundamental concepts of computer networking, protocols, architectures, and applications.
- Understand the division of network functionalities into layers.
- Learn the flow control and error control mechanisms.
- Obtain the skills of subnetting and routing mechanisms.
- Explain the transport layer and application layer protocols.

D.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No’s	Course Outcomes	K - Level
CO1	Explain the basic fundamentals of networks and discuss the various techniques and standards of analog and digital data communication.	K2
CO2	Apply various error detection, correction and flow control techniques for efficient transmission of data.	K3
CO3	Interpret the functions and protocols of network layer.	K3
CO4	Explain the functions and protocols of transport layer.	K3
CO5	Understand the various protocols of application layer	K2
Knowledge Level (Based on revised Bloom’s Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of COs with Program outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2		2				2				2	
CO2	3	2		2				2				2	
CO3	3	3		2				2				3	
CO4	2	3		2				2				2	
CO5	3	2						2					

3- High; 2-Medium; 1-Low

F. Course Contents

Unit 1 Introduction and Physical Layer L-9 Hours

ISO/OSI - TCP/IP Protocol suite, Data Communication-system components – Topology-Transmission ImpairmentTransmission Media: Guided Media - Unguided Media - Wireless - Wired LANs: Ethernet - Token ring - Connecting Devices – Switching techniques – Bandwidth Metrics.

Unit 2 Data Link Layer L-9 Hours

Types of errors –Error detectionand correction techniques, Flow control mechanism, Multiple access protocols, Link layer addressing. Media Access Control – frame format, types, cloning. IEEE 802.3 – frame format, types.

Unit 3 Network layer L-9 Hours

Logical Addressing: Addressing Methods –Classfull addressing - IPV4 – Public and private addressing.Classless addressing - IPV6 - Frame format –Subnetting -Supernetting, CIDR.Inter/Intra networks,ARP– DHCP – ICMP – IGMP - SNMP. Unicast Routing Protocols – Intra and Interdomain Routing - Distance Vector Routing - Link State Routing. RIP, OSPF.

Unit 4 Transport Layer L-9 Hours

Transport Layer: Process-to-Process Delivery: UDP – TCP - Congestion Control and Flow control Techniques - Quality of Service - Techniques to Improve QoS, merits and de-merits.

Unit 5 Application Layer L-9 Hours

Session Layer: Functions-Protocols: Winsock, Socket-Remote Procedure Call-Presentation Layer: Functions- Compression Techniques: Lossy and Lossless - Application layer protocols: REMOTE LOGGING – TELNET/SSH -EMAIL– DNS – FTP/TFTP– IPSec - HTTP/HTTPS – SDN.

Total: 45 Hours

G. Learning Resources

i. Text Books:

1. Behrouz Forouzan, "Data Communications and Networking", Tata McGraw Hill, 5th Edition,2021. (Reprinted) (Unit 1 to 5)

ii. Reference Books:

1.Kurose and K.W. Ross, "Computer Networking: A Top-Down Approach Featuring the Internet", Addison-Wesley,2017

2. Andrew S. Tanenbaum, David J Wetherall, Computer Networks, 5th Edition, Pearson Edu, 2013.

3. Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud,5th Edition,2017

iii. Online References:

1."Computer Communication Specialization" Course Era [online] 2019. Available:https://www.coursera.org/specializations/computer-communications

2."Introduction to Networking" edx [online]2020,Available: https://www.edx.org/course/introduction-to-networking

3."Computer Networking"Udacity [online] 2020, Available: https://www.udacity.com/course/computer-networking--ud436

4."Computer Networks and Internet Protocol" NPTEL [online] 2018, Available: https://nptel.ac.in/courses/106/105/106105183/

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC106	Formal Languages and Automata Theory	3	0	0	3

A.Preamble

This course deals with the concepts of automata theory, formal languages, grammar, computability and decidability. Students learn concrete implementations, manipulations of discrete structures and their use in design and analysis of non-trivial problems for a given computational task. This course can be applied in Compilers, -Artificial Intelligence, Robotics and Natural Language Processing.

B.PrerequisiteCourse

10210MA110-Discrete Mathematical Structures

C.CourseObjectives

Learners are exposed to

- Understand overview of theoretical foundations of computer science from the perspective of formal languages.
- Illustrate various automata to solve problems in computing.
- Familiarize Regular grammars, context free grammar, recursive and recursively enumerable languages.

D.CourseOutcomes

Uponthesuccessful completionof thecourse,students will beableto:

CO No’s	CourseOutcomes	K -Level
CO1	Design the finite automata to recognize the regular languages.	K3
CO2	Construct regular expression for regular grammar and its equivalence with finite automata.	K3
CO3	Develop push down automata and context-free grammar representations for context-free languages.	K3
CO4	Model Turing Machines for accepting recursive languages and its capabilities.	K3
CO5	Apply the notions of decidability and undecidability to examine complex problems.	K3
KnowledgeLevel(Based onrevised Bloom’sTaxonomy) K1-RememberK2-UnderstandK3-ApplyK4-AnalyzeK5-EvaluateK6-Create		

E.CorrelationofCoswithProgramoutcomesandProgrammeSpecificOutcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	3		3								
CO2	3	3	3		3								
CO3	3	3	2	1	3								
CO4	3	3	3	3	3								
CO5	3	3	2	2									

3-High; 2-Medium; 1-Low

F.CourseContents

Unit 1 Formal Languages and Finite Automata L–10 Hours

Introduction to automata and automata theory, Basic concepts of formal Languages, Finite Automata – DFA, NFA, Epsilon NFA, Equivalence of DFA, NFA and Epsilon NFA, Minimization of Automata, FA with output. Case Studies: Implementation of Spell checkers, Text search, Text editors, Simulation of FA using JFLAP.

Unit 2 Regular Languages and Expressions L–8 Hours

Regular Grammar, Regular Expressions, Converting Regular Expression to Epsilon NFA, Equivalence of regular expressions and NFA with Epsilon moves, Converting DFA to Regular Expressions, Pumping Lemma for Regular Languages, Applications of Pumping Lemma, Closure Properties of Regular sets. Case Studies: Pattern for Mobile numbers with country code, email address

Unit 3 Context Free Languages and Push Down Automata L–10 Hours

Context-Free Languages and Grammar, Derivation trees, Ambiguity, Simplification of CFG, Chomsky Normal Form, Greibach Normal Forms, Deterministic Push Down Automata, Non-Deterministic Pushdown Automata (NPDA), Equivalence of acceptance by final state and empty stack in PDA, Equivalence between NPDA and CFG, Closure properties of CFLs, Pumping Lemma for CFLs. Case Studies: Tower of Hanoi, Evaluating Arithmetic expression, Transaction Process System.

Unit 4 Turing Machine L–9 Hours

Context-sensitive Grammar, Turing Machine (TM) – Basics and formal definition, Instantaneous Description, TMs as language acceptors, TMs as Transducers, Designing Turing Machines, Variants of TMs, Checking of Symbols, Encoding a Turing Machine and Universal Turing Machine. Case Study: Lambda Calculus

Unit 5 Recursive and Recursively Enumerable Languages L–8 Hours

Recursive Functions, Recursive languages and Recursively Enumerable Languages, Properties of Recursively Enumerable Languages and Recursive Languages, Decidability and Halting Problem, Reduction, P and NP, NP- completeness, Post Correspondence Problem, Rice Theorem and Chomsky Hierarchy.
Case Studies: Knapsack Problem, SAT Problem using Turing machine.

Total: 45 Hours (L)

G. Learning Resources

i. Text Books:

1. John E Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, “Introduction to Automata Theory, Languages, and Computation”, 3/e, Pearson Education, 2013. [Unit 1 – 3]
2. John C. Martin, “Introduction to Languages and the theory of computation”, 4/e, TMH, 2011. [Unit 4, 5]

ii. Reference Books:

1. Michael Sipser, “Introduction to the Theory of Computation”, 3rd edition, Cengage Publishers, 2013.
2. Peter Linz, “An Introduction to Formal Language and Automata”,5th edition,Cathleen Sether Publishers, 2012.
3. S.P.Eugene Xavier, “Theory of Automata, Formal Languages and Computation”, New Age International Publishers,2005.

iii. Online References:

1. “Theory of Computation” Sep. 09, 2016. Accessed on: Feb. 16, 2021, [Online]. Available: https://nptel.ac.in/courses/106/104/106104148/
2. “Automata Theory”, Accessed on: Apr. 21, 2021, [Online]. Available: https://www.edx.org/course/automata-theory
3. “Lambda Expressions” [Online] Accessed on: Apr. 28, 2021, https://www.udemy.com/course/java-8-functional-programming-with-lambda-expression/

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC107	**Compiler Design**	**3**	**0**	**0**	**3**

A. Preamble

This course introduces the basic concepts and mechanisms that are employed in language translators, with emphasis on compilers. It aims to provide the theory and practice of compilation, in particular, the lexical analysis, parsing, intermediate code generation, techniques of code optimization and approaches towards code generation.

B. Pre-requisite Course

10211CC106 - Formal Languages and Automata Theory

C. Course Objectives

Learners are exposed to

- Understand the structure of a compiler with the sequence of translation of input for each phase.
- Analyze the regular languages to describe the lexical elements of a programming language.
- Develop the syntactic structure of a programming language using the context free language and design the top-down and bottom-up parsers.
- Break down the statements and expressions to form and implement the intermediate code.
- Optimize the intermediate code of a program to machine-specific instructions to generate target code

D. Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Interpret the different phases of compilation with compile time error handling and design the lexical analyzer for a language.	**K3**
CO2	Develop appropriate top down and bottom-up parser to produce parse tree representation of the input.	**K3**
CO3	Generate intermediate code for decision control and looping control statements in high level language.	**K3**
CO4	Apply various optimization techniques with basic blocks of the intermediate code through global data flow analysis.	**K3**
CO5	Design DAG representation for basic blocks and target low level language for high level language.	**K3**
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of Cos with Program outcomes and Programme Specific Outcomes:

Cos	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	1		2							2	
CO2	3	2	1		1							2	
CO3	3	3										1	2
CO4	3	3										1	
CO5	3	3	1									2	

3- High; 2-Medium; 1-Low

F. Course Contents

Unit 1 Compilers and Lexical Analysis L–9 Hours

Compilers – Interpreters – Analysis of the source program –Cousins of the Compiler – Grouping of Phases – Phases of a compiler – Compiler construction tools – Lexical Analysis – Role of Lexical Analyzer – Input Buffering – Recognition of Tokens – Specification of Tokens. Design of a Lexical Analyzer Generator. Case Study: Lexical Analyzer for separating tokens, counting whitespace, special characters and newline character, Pattern Matching of strings – use relevant tool.

Unit 2 Syntax Analysis and Parsers L–9 Hours

Need and Role of the parser –Writing Grammars – Context-Free Grammars – Top Down parsing – Recursive Descent Parsing – Predictive Parsing – Bottom-up parsing – Shift Reduce Parsing – Operator Precedence Parsing – LR Parsers – SLR Parser – Canonical LR Parser – LALR Parser. Case Study: Specification for desktop calculator, Error recovery of ambiguous arithmetic grammar, Syntax Analyzer for looping construct - use relevant tool.

Unit 3 Intermediate Code Generation L–9 Hours

Intermediate languages – Implementation of Three Address Code – Types and Declarations – Type Checking – Assignment Statements – Boolean Expressions – Switch Case Statements – Back patching – Procedure calls. Case Study: Intermediate code for Array references with subscripts, Intermediate code for decision making and looping constructs of C, Intermediate code for Expression Parsing.

Unit 4 Code Optimization L–9 Hours

Principal Sources of Optimization – Introduction to Data Flow Analysis – Constant propagation – Loops in Flow Graphs – Runtime Environments – Source Language issues – Storage Organization – Storage Allocation – Access to Nonlocal data - Local optimization-Global optimization. Case Study: Implement code optimization methods using relevant data structures.

Unit 5 Code Generation L–9 Hours

Issues in the design of code generator – The target machine – Register Allocation and Assignment- Basic Blocks and Flow Graphs – Optimization of basic Blocks – A Simple Code Generator – DAG representation of Basic Blocks – Code Generation from DAG - Heuristic Reordering - Labeling - Next-use Information – Peephole Optimization. Case Study: Graph coloring application using Register Allocation, Generate Target code for Binary tree operations.

Total: 45 Hours (L)

G. Learning Resources

i. **Text Book:**

1. Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman, "Compilers – Principles Techniques and Tools, Pearson", Second Edition, 2013. [Unit 1-5]

ii. **Reference Books:**

1. Kenneth C. Louden, "Compiler Construction –Principles and Practice", Cengage Learning Indian Edition, 2006.
2. Grune D, Bal H, Jacobs C and Langendoen K, "Modern Compiler Design", Wiley, 2000.
3. Cooper K and Torczon L, "Engineering a Compiler", Morgan Kaufmann, Second Edition, 2011.

iii. **Online References:**

1. "COMPILER DESIGN" Sep. 25, 2014. Accessed on: Feb. 16, 2021 [Online]. Available: https://nptel.ac.in/courses/106/105/106105190/
2. "LEX & YACC TUTORIAL by Tom Niemann" Oct. 24, 2013. Accessed on: Apr. 20, 2021 [Online]. Available: https://cse.iitkgp.ac.in/ ~bivasm/ notes/ LexAndYacc Tutorial.pdf
3. "Compiler Design", Nov. 2019. Accessed on: Apr. 20, 2021 [Online]. Available: https://www.udemy.com/course/compiler-design-n/

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC109	Microprocessors	2	0	0	2

A. Preamble

Microprocessors are a core course in the stream of Computer Science & Engineering. The objective of this course is to explore the concepts of hardware and software part of a microprocessor. It includes the architecture of 8086 Microprocessor and latest ARM processor architecture. This course covers ARM processor programming with interfacing techniques and equivalent assembly language code for high level language instructions like ‘C’.

B. Pre-requisite Course

10210EC201 – Basic Electronics & Digital Logic Design

C. Course Objective

Learners are exposed to:

- Understand the architecture of 8086.
- Impart the knowledge about the instruction set.
- Understand the basic idea ARM processor architecture and its applications.

D. Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No’s	Course Outcomes	K - Level
CO1	Know the architecture of 8086 microprocessor and its basic configuration.	K2
CO2	Develop assembly language programs for the given problems using 8086 processor.	K3
CO3	Understand ARM processor architecture and the principle of its debug system.	K2
CO4	Develop programs for the problems specified using ARM processor instructions.	K3
CO5	Establish interfacing of I / O devices with ARM processor to develop real time applications.	K3
Knowledge Level (Based on revised Bloom’s Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of COs with Program outcomes and Programme Specific Outcomes:

Cos	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3											
CO2	3	2			3							2	
CO3	3	2											
CO4	3	2			3							3	
CO5	3		3		3							3	

3- High; 2-Medium; 1-Low

F. Course Contents

Unit 1 Microprocessor Architecture – 8086 **L-6 Hours**

Introduction – Types of computers – Evolution of Microprocessor – 8086 Microprocessor architecture – 8086 Programming method –– program development tools – Interrupts and interrupt service procedures - 8086 signals – Basic configurations – System connections – Timing – Troubleshooting – Assembler directives - Introduction to Open RISC Architecture.

Unit 2 Programming In 8086 **L-6 Hours**

Assembly language programming – Addressing modes – Instruction set: Data transfer instructions – arithmetic and logical instructions – Branch instructions –– Loop instructions – While – Do programs – Strings – Procedures – Macros.

Unit 3 Arm Processor Architecture **L-6 Hours**

Cortex – M architecture – Block diagram –pipelined architecture – memory address map – bus system and bus matrix – memory and peripherals: memory endianness – bit banding – system stack architecture – debug system: AHB access port, FPB, DWT, ITM, ETM, TPIU – conditional execution – interrupts: exception and interrupt handling.

Unit 4 Arm Processor Programming **L-6 Hours**

Addressing modes – ARM instruction sets: thumb instruction set – ADD and MOV data processing instructions – LDR and STR memory access instructions – data processing instructions – bit field instructions – branch and control instructions – recursive functions If-Then conditional instructions –reset sequence – instruction encoding.

Unit 5 Arm Processor Interfacing **L-6 Hours**

TM4C123 microcontroller block diagram – GPIO features – peripherals on the memory map – GPIO pin configuration – LED and switch interfacing – keypad interfacing – I/O synchronization – polling – interrupt driven – processor interrupt configuration – interrupt based interfacing – Timer: timer as I/O device – external event counter – input capture device – general purpose timer modules – serial communication interface – I^2C interface – SPI interface – CAN interface.

Total: 30 Hours (L)

G. Learning Resources

i. Text Books:

1. Doughlas V. Hall, "Microprocessors and Interfacing", TMH, Revised second edition, 2005. [Unit 1-2]
2. Muhammad Tahir, Kashif Javed, "ARM Microprocessor Systems: Cortex-M Architecture, Programming, and Interfacing", CRC Press, Taylor & Francis group, 2017. [Unit 3-5]

ii. Reference Books:

1. Andrew N. Sloss, "ARM system developer's guide – Designing and optimizing system software", ELSEVIER, 2004.
2. Charles M. Gilmore, "Microprocessors: Principles and applications", TMH, Second edition, 1995.
3. Jonathan W. Valvano, "Embedded microcomputer systems: Real time interfacing", CENGAGE Learning, Third edition, 2012.
4. Steve Furber, "ARM system-on-chip architecture", Pearson education, Second edition, 2015.

iii. iii. Online References:

1. "Microprocessor interfacing", Accessed on Sep 20, 2022 [Online], Available: https://onlinecourses.nptel.ac.in/noc20_ee11/preview.
2. "Microprocessors and Digital systems", Accessed on Sep 21, 2022 [Online], Available: https://archive.org/details/microprocessorsd0000hall/page/n3/mode/2up.
3. "ARM processor architecture", Accessed on Sep 21, 2022 [Online], Available: https://www.cs.ccu.edu.tw/~pahsiung/courses/ese/notes/ESD_03_ARM_Architecture.pdf.

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC119	Cryptography and Network Security	3	0	0	3

A. Preamble

The purpose of this course is to understand principles of encryption algorithms, conventional and public key cryptography. It also includes knowledge of authentication, hash functions and also application level of security mechanisms.

B.Prerequisite Course

10211CC130- Fundamentals of Computer Networks

C.Course Objectives

Learners are exposed to

- Understand the basics of cryptography, its techniques and principles.
- Make use of mathematical concepts to apply the public key cryptography.
- Utilize the authentication hash functions and digital signatures for security.
- Organize various cryptographic standards to apply over security application.
- Identify the preventive security mechanisms for efficient security management.

D.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No’s	Course Outcomes	K - Level
CO1	Understand the encryption and decryption techniques using block ciphers.	K2
CO2	Apply key exchange and management schemes using public key cryptography.	K3
CO3	Demonstrate techniques to sign and verify messages using well known signature generation and verification algorithms.	K3
CO4	Implement the cryptographic algorithmfor various network security applications.	K3
CO5	Identify the technologies to protect cipher space against security threats.	K3
Knowledge Level (Based on revised Bloom’s Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of COs with Program Outcomes and Program Specific Outcomes

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3										2	3
CO2	3	2		3		2						3	3
CO3	3	2		3		2						3	3
CO4	2				1	2						1	2
CO5	2	2		2	2							2	2

3- High; 2-Medium; 1-Low

F.Course Contents

Unit 1 Introductionto Cryptography L-9 Hours

OSI Security Architecture - Classical Encryption techniques – Cipher Principles – Data Encryption Standard – Block Cipher Design Principles and Modes of Operation - Evaluation criteria for AES –AES Cipher – Triple DES – Placement of Encryption Function – Traffic Confidentiality-Case study on Barclays Bank.

Unit 2 Public Key Cryptography L-9 Hours

Number Theory concepts: Primes and Prime Factorization –Congruent modulo n, equivalent class modulo n, Integer modulo n, Multiplicative inverse, Relatively prime, Euler's theorem, Fermat's little theorem, Extended Euclidean Algorithm, Chinese Remainder Theorem. Confidentiality using Asymmetric Encryption – Public Key Cryptography and RSA- Key Management - Diffie-Hellman key Exchange – Elliptic Curve Architecture and Cryptography – Case study on Elan Financial Services

Unit 3 Authentication and Hash Function L-9 Hours

Authentication requirements – Authentication functions – Message Authentication Codes – Hash Functions – Security of Hash Functions and MACs – MD5 message Digest algorithm - Secure Hash Algorithm – RIPEMD – HMAC Digital Signatures – Authentication Protocols – Digital Signature Standard- Case study on Swedbank

Unit 4 Network Security Applications L-9 Hours

Authentication Applications: Kerberos – X.509 Authentication Service – Electronic Mail Security –PGP – S/MIME - IP Security- Policy, Encapsulating Security Payload, Combining Secrity Associations, Internet Key Exchange, Authentication Header.

Unit 5 Security Management L-9 Hours

Intrusion detection and Prevention System – password management – Viruses and related Threats – Virus Countermeasures- Worms Security Risks – Firewall Design Principles – Trusted Systems- Log Management. Case study on Biometric deployment for secure password management.

Total: 45 Hours (L)

G.Learning Resources

i.Text Books:

1. William Stallings, “Cryptography And Network Security – Principles and Practices”, Prentice Hall of India, Eighth Edition, 2020.[Unit 1-5]
2. David Kim and Michael G.Solomon, “Fundamentals of Information Systems Security”, Jones and Bartlett Publishers, Third Edition, 2018.

ii.Reference Books:

1. Atul Kahate, “Cryptography and Network Security”, Tata McGraw-Hill, 2011.
2. Bruce Schneier, “Applied Cryptography”, John Wiley & Sons Inc, 2011.
3. Charles B. Pfleeger, Shari Lawrence Pfleeger, “Security in Computing”, Third Edition, Pearson Education, 2010.

iii.Online References:

1. “Cryptography techniques”, Accessed on: July 5 2021, [online]. Available: http://Cryptographywilliamstallings.com/Extras/Security-Notes/
2. “Authentication algorithms”, Accessed on: July 5 2021 [online]. Available: http://www.cs.bilk.ent.edu.tr/~selcuk/teaching/cs519/
3. “Network security concepts”, Accessed on: July 5 2021[online]. Available:http://freevideolectures.com/Course/3027/Cryptography–andNetwork-Security

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC202	**Design and Analysis of Algorithms**	**3**	**0**	**2**	**4**

A.Preamble

The course introduces the basics of computational complexity analysis and various algorithm design paradigms. The goal is to provide students with solid foundations to deal with a wide variety of computational problems, and to provide a thorough knowledge of the most common algorithms and data structures.

B.Prerequisite Course

10211CC101 – Data Structures

C.Course Objectives

Students are able to

- Develop the problem-solving ability and algorithmic analytical skills.
- Produce optimized code.
- Understand the different techniques involved in solving a problem.
- Analyze the logic in terms of Efficiency, Time & Space Complexity.
- Break the large size algorithm into smaller size components.

D.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Analyze the time complexity ofalgorithms using various asymptotic notations.	**K3**
CO2	Implement solution for the given problems using Brute force and Divide &Conquer techniques.	**K3**
CO3	Apply Dynamic programming and Greedy techniques to solve the problems.	**K3**
CO4	Solve the given problems using Backtracking and Branch & Bound techniques.	**K3**
CO5	Interpret the computational efficiency using iterative approaches.	**K2**
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of COs with Program outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	3	3				1	1		1	2	
CO2	3	3	3	3				1	1		1	2	
CO3	3	3	3	3				1	1		1	2	
CO4	3	3	3	3				1	1		1	2	
CO5	3	3	3	3				1	1		1		

High- 3; Medium-2; Low-1

F.Course Contents

Unit 1 Introduction L–9 Hours

Notion of an Algorithm – Fundamentals of Algorithmic Problem Solving – Important Problem Types – Analysis of Algorithm Efficiency - Introduction to Asymptotic Notations– Solving Recurrence relations – Master's theorem - Mathematical analysis for Recursive and Non-recursive algorithms.

(**Case Study:** Design a gaming application like Chess, Candy crush etc.)

Unit 2 Brute Force and Divide & Conquer L–9 Hours

Brute Force: Closest-Pair and convex-Hull Problems- Exhaustive Search - Traveling Salesman Problem - Knapsack Problem - Assignment problem. Divide and conquer methodology: Merge sort – Quick sort – Binary search- Heap sort- Closest-Pair and convex-Hull Problems

(Case Study: Brute Force: Solve the hide and seek game with a search time limit as 5 minutes and check its efficiency.)

(Case Study: Divide & Conquer: A king witha huge kingdom struggles to oversee from a single central city. Suggest how the king will handle the local problems of the smaller parts of the kingdom.)

Unit 3 Dynamic Programming and Greedy Technique L–9 Hours

Dynamic Programming: Principle of optimality –Rod Cutting Problem - Computing Binomial Coefficient - Warshall's and Floyd' algorithm – Knapsack Problem. Greedy Technique: Minimum spanning tree-Prim's algorithm- Kruskal's Algorithm- Huffman Trees and Codes- Graph algorithms: Topological sort- Dijkstra's Algorithm.

Case Study: Dynamic Programming: Hill Climbing,Comparison of Knapsack Problem with other design algorithms, A Taxi Operator firm needs to design a growth strategy plan that should focus on the revenue for the next 25 years considering various constraints like car turns old, reduction in mileages etc.,)

(Case Study: Greedy Technique: Assumethe list of arrival and departure time; find the minimum number of platforms required for the railway as no train waits.)

Unit 4 Backtracking and Branch & Bound L–9 Hours

Backtracking: n-Queens problem- Hamiltonian Circuit Problem- Subset sum problemBranch & Bound: Assignment problem-Knapsack Problem- Traveling Salesman Problem

(Case Study: Backtracking: Solve a constraint network to find an assignment of values for each variable so that all constraints are satisfied.)

(**Case Study: Branch & Bound:** Optimize the tour to deliver the packages at randomly chosen pizza centres in the city of Chennai.)

Unit 5 Iterative Improvements and Limitations of Algorithm Power L–9 Hours

The simplex method– The maximum-flow problem– The Maximum matching in bipartite graph- Introduction to unsolvable problem-Halting problem. Limitations of Algorithm

Power--Decision Trees- Lower bound arguments- P, NP and NP-Complete Problems. The Classes NP and P.

(**Case Study:** Identify solution to the telecommunication networks to handle the crucial problems such as flow control and routing)

Total: 45 Hours (L)

G.Laboratory Experiments

Task 1: Algorithmic problem solving 1

Calculating the time complexities for the given algorithms and estimating the orders of growth

Ex 1:

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
   for (j = 2; j <= n; j = j * 2) {
      k = k + n / 2;
   } }
```

Ex 2:

```
function(int n) {
   if (n==1)
     return;
   for (int i=1; i<=n; i++)
   {
      for (int j=1; j<=n; j++)
      {
         printf("*");
         break;
      }    }
```

Task 2: Algorithmic problem solving 2

Ex 2.1: Sort all the functions below in increasing order of asymptotic (Big O) growth. If some have the same asymptotic growth, then be sure to indicate that. Note: log means base 2.

i) $5n$, ii) $4 \log n$, iii) $4 \log \log n$, iv) n^4, v) $n^{1/2} \log^4 n$, vi) $(\log n)^{5\log n}$ vii) $n^{\log n}$, viii) 5^n, ix) 4^{n4}, x) 4^{4n} xi) 5^{5n}, xii) 5^{5n}, xiii) $n^{n1/5}$, xiv) $n^{n/4}$, xv) $(n/4)^{n/4}$

Solution: $4 \log \log n < 4 \log n < n^{1/2} \log^4 n < 5n < n^4 < (\log n)^{5\log n} < n^{\log n} < n^{n1/5}$
$< 5^n < 5^{5n} < (n/4)^{n/4} < n^{n/4} < 4^{n4} < 4^{4n} < 5^{5n}$

Ex 2.2 Order the functions according to their growth from slowest to fastest growing

Ex2.3: $6n^3$, $n \log_6 n$, $4n$, $8n^2$, $\log_2 n$, $n\log_2 n$, $\log_8 n$, 64, 8^{2n}

Solution:64, $\log_8 n$, $\log_2 n$, $4n$, $n \log_6 n$, $n\log_2 n$, $8n^2$, $6n^3$, 8^{2n}

Task 3: D&C – Quick Sort

Sort a given set of elements using the Quicksort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

Test Case 1: Choose random element as Pivot and analyze the time taken to execute the algorithm

Test Case 2: Assume that the list of elements is in ascending order and apply quick sort and find the time complexity

Task 4: D&C – Merge Sort

Implement a parallelized Merge Sort algorithm to sort a given set of elements and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

Test Case 1: Assume that the list of elements is in ascending order and apply merge sort and find the time complexity
Test Case 2: Assume the unsorted order of elements and apply merge sort and find its time complexity. Modify the algorithm to arrange the elements in descending order.

Task 5: Dynamic Programming – Warshall's & Floyd's algorithm

Compute the transitive closure of a given directed graph using Warshall's algorithm and for the same graph implement the all pairs shortest path problem using Floyd's algorithm.

Test Case 1: Assume the graph or a digraph with negative weights and check whether the Floyd's algorithm yields the correct result or not.
Test Case 2: Analyze that the time efficiency of Warshall's algorithm is cubic.

Task 6: Dynamic Programming – Knapsack algorithm

Implement 0/1 Knapsack problem using Dynamic Programming and find its complexity

Test Case 1: Show that a sequence of values in a row of the dynamic programming table for the knapsack problem is always nondecreasing
Test Case 2: Show that a sequence of values in a column of the dynamic programming table for the knapsack problem is always nondecreasing

Task 7: Greedy Technique – Prim's algorithm

Find Minimum Cost Spanning Tree of a undirected graph using Prim's algorithm

Test Case 1: Assume an undirected graph and find whether the graph is connected or not and then apply Prim's algorithm to find MST.
Test Case 2: Verify whether Prim's algorithm always work correctly on graphs with negative edge weights.

Task 8: Greedy Technique – Kruskal's algorithm

Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm

Test Case 1: Assume an input graph with more than one edge having a same weight, in such case which edge should be selected as the next visiting vertex by applying Kruskal's algorithm.

Test Case 2: Construct a Maximum Spanning tree for the input graph by applying Kruskal's algorithm.

Task 9: Greedy Technique – Dijkstra's algorithm

From a given vertex in a weighted connected graph find shortest paths to other vertices using Dijkstra's algorithm

Test Case 1: Show that the Dijkstra's Algorithm cannot work for a weighted connected graph with negative weights.
Test Case 2: Analyze the correctness of Dijkstra's algorithm for graphs with positive weights.

Task 10: Greedy Technique – Topological Sort

Obtain the Topological ordering of vertices in a digraph.

Test Case 1: Assume a Directed Acyclic Graph (DAG) as an input and check whether Topological sorting can be applied or not?
Test Case 2: Implement topological sorting using Breadth First Search (BFS)

Task 11: Backtracking - n-Queen's problem

Implement N Queen's problem using Backtracking

Test Case 1: Implement Rat in a Maze problem by applying the concept of backtracking.
Test Case 2: Modify the n-queens problem to handle the value of n in the range from 1 to 3.

Task 12: Backtracking - Subset problem

Find a subset of a given set S = {s1,s2,.....,sn} of n positive integers whose sum is equal to a given positive integer d. For example, if S= {1, 2, 5, 6, 8} and d = 9 there are two solutions {1,2,6} and {1,8}.A suitable message is to be displayed if the given problem instance doesn't have a solution.

Test Case 1: Assume an instance of the subset sum problem and verify whether the backtracking algorithm work correctly if we use just one of the two inequalities to terminate a node as non-promising.
Test Case 2: Given a set of distinct integers and return all the possible subsets or power set.

Task 13: Branch & Bound - Travelling Salesman problem

Implement any scheme to find the optimal solution for the TravellingSalesperson problem

Test Case 1: Derive the solution of TSP using B&B and prove that it is optimal when compared with Brute Force technique.
Test Case 2: Analyze that the Travelling Salesman Problem (TSP) is an NP hard problem.

Task 14: Branch & Bound – Knapsack algorithm

Implement 0/1 Knapsack problem using Branch & Bound techniques and find its complexity

Test Case 1: Derive the solution of Knapsack problem using B&B and prove that it is optimal when compared with Brute Force technique.
Test Case 2: Show that a sequence of values in a column of the dynamic programming table for the knapsack problem is always non decreasing.

Task 15: Iterative Improvements

Apply Ford-Fulkerson algorithm to solve the computational problems of finding maximum flow in a network to determine how fast people can be evacuated from the given city.
Test Case 1: Apply Ford-Fulkerson algorithm and find whether the augmenting path exists or not in the graph.
Test Case 2: Apply Ford-Fulkerson algorithm and find the updated residual capacity of a graph.

Total: 30 Hours

LIST OF EQUIPMENT FOR A BATCH OF 30 STUDENTS:

Stand alone systems with C/C++ compiler 30 Nos.

(or)

Server with C/C++ compiler supporting 30 terminals or more.

H. Learning Resources

i. Text Books:

1. Anany Levitin, “Introduction to the Design and Analysis of Algorithms”, Third Edition, Pearson Education, 2012. [Unit 1, 2, 4]
2. E.Horowitz, S. Sahni, and S. Rajasekaran, “Computer Algorithms,” 2nd Edition, Galgotia Publications, 2007. [Unit 3, 5]

ii. Reference Books:

1. Alfred V. Aho, John E. Hopcroft and Jeffrey D. Ullman, “Data Structures and Algorithms”, Pearson Education, Reprint 2017.
2. Thomas H.Cormen, Charles E.Leiserson, Ronald L. Rivest and Clifford Stein, “Introduction to Algorithms”, Third Edition, PHI Learning Private Limited, 2012.
3. Donald E. Knuth, “The Art of Computer Programming”, Volumes 1&3 Pearson Education, 2009.

iii. Online References:

1. “Design and analysis of algorithms”, Jan 21 2015, Accessed on: Jan 18 2021, [Online]. Available: https://onlinecourses.nptel.ac.in/noc21_cs22/preview.
2. "Algorithms Specialization", Accessed on: April 21 2021, [Online], Available: https://www.coursera.org/specializations/algorithms.

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC204	Programming Using Java	3	0	2	4

A.Preamble

This course provides basic concepts about Object Oriented Programming, Java Packages, Exceptions, Database connectivity, Java Streams, JavaFX and Java Servlets, JSP, Spring and Hibernate. After successful completion of this course learners can able to develop software modules for real world problem.

B.Pre-requisite course

10210CS101–Problem solving using C

C.Course Objectives

Learners are exposed to

- Implement Basic OOP principles to solve problems.
- Practice Java basic constructs.
- Build efficient Programming practice in Java.
- Web application development with DB servers.
- Work on Frameworks such as Spring and Hibernate.

D.Course Outcomes:

Upon the successful completion of the course, students will be able to:

CO No’s	Course Outcomes	K - Level
CO1	Explain the fundamentals of Java and Object Oriented Programming (OOP) concepts to solve problems.	K2
CO2	Apply the concepts of java inheritance and exception handling mechanisms for real world problem.	K3
CO3	Solve the problems using Java Streams, Collections, UI Design and Threads.	K3
CO4	Develop simple web applications using JSP, Servlet with JDBC and Hibernate Framework.	K3
CO5	Build the web applications using Spring Framework.	K3
Knowledge Level (Based on revised Bloom’s Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of COs with Program outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	1										
CO2	1	3	3	1	1							2	1
CO3	1	3	2	2	2							2	2
CO4	1	3	3		2								3
CO5	1	3	3		3								3

3-High; 2-Medium; 1-Low

F.Course contents

Unit 1 Basics of OOP and JAVA L-9 Hours

Introduction to object-oriented programming - Features of Java - JVM- Keywords- Variables - Data types – Operators - Control statements -Command line arguments - Classes and Methods - Objects –Java Packages – Build-in packages - user-defined package -Access specifiers: private, protected, default, public - Constructors - Method Overloading - Type Casting - this keyword- Static – Arrays, Multi-dimensional Arrays.

Unit 2 Inheritance, Exception Handling and String L-9 Hours

Basics of Inheritance - Forms of Inheritance - Super keyword – Final - Method Overriding – Abstraction: Abstract Classes - Interfaces.
Exception Handling: Java Exception Hierarchy - Exception Types - Throwing and Catching exceptions -Declaring New Exception Types – custom exceptions.
String class – Immutable String - String handling methods.

Unit 3 Java Streams, Collections, UI Design, Thread L-9 Hours

Java Streams: Features, methods – Generic Classes and Methods - Collections Frameworks: List, Queue, Set and Map.
JavaFX: Architecture, Effects, Transformation, Animation, Layouts, UI Controls, Event Handling.
Threads: Life cycle - Thread states - Thread priority– Thread Synchronization.

Unit 4 Server Programming, Hibernate Framework L-9 Hours

Introduction to Servlet and JSP, JDBC, Example Web applications with DB - Basics of Java Networking.
Introduction to Java Frameworks - Basics of Hibernate – Hibernate Web application - Hibernate Mapping and types – HQL – Annotations – Caching. Integrating Hibernate and Spring.

Unit 5 Java Spring Framework L-9 Hours

Basics of Spring – Spring in IDE– Dependency Injection – Constructor Injection – Spring ORM: Spring with Hibernate – SPEL -Spring MVC – Spring Web – Spring JDBC – Spring with Angular – Spring Boot.

Total: 45 Hours

G.Laboratory Experiments

PART – 1:

Task 1: Package creation and String.

- Simple Java application with class (Object), Methods
- Developing user defined packages in Java
- Apply String handling methods

Task 2: Inheritance and Interfaces

- To Implement Method overloading, constructor and Method Overriding.
- Use of this, super, static and final keywords
- Developing user-defined interfaces and implementation
- Use of predefined interfaces

Task 3: Exception Handling Mechanism in Java

- Handling pre-defined exceptions
- Handling user-defined exceptions

Task 4: Java Collections

- Handle list of elements using Collection classes (List, Queue, Set, Map)

Task 5: JavaFX

- To Create Different Layouts with JavaFX animation.
- To Design a web application with JavaFX event handling.

Task 6: Threading

- Creation of thread in Java applications
- Multithreading

Task 7: Java-JDBC

- To connect Oracle/ MySQL for Table creation and Data Manipulation.

Task 8: Servlet

- Write a Servlet program to connect Oracle/MySQL for Table creation and Data Manipulation.

Task 9:JSP

- Write a JSP program to connect Oracle/ MySQL for Table creation and Data Manipulation.

Task 10: Spring Standalone and web Application

- Standalone application with data manipulation
- Web application with data manipulation

Task 11: Hibernate Standalone and web Application

- Standalone application with data manipulation
- Web application with data manipulation

PART – 2:
Use cases:

Use Case-1:Validated Login for Employee management JSP:

Company ABC planned to create a website for maintaining employee details. Login page need to be designed with username, password and captcha. Username is their mail-id and password should contain at-least 1-Uppercase alphabet, 1-special character, 1- lower alphabet. Design the page using front end and server scripting with client / server validation.
Use the technologies: HTML5, CSS5, Javascript, JSP.

Use Case-2:Event Registration for a University Spring:

In a University, they have planned to conduct national level symposium. Students are expected to come from various places from india. University has decided to maintain all participant details such as Participant_name, college name, branch, year, age, and phone. And also generate unique participant ID using ID generator logic. Implement a java application for above scenario with rich user interface and database server.
Use the technologies: HTML5, CSS5, Spring framework, MySQL/Oracle.

Use Case-3:E – Applicant System for Student using Spring MVC:

E-Applicant system has been planned to implement by an institute. In that, student will be able to request any kind of Certificate like Transfer and Bonafiedetc by applying

through this System.The student can check the status through his/her account.There are many users like Internal Staff, HOD, Dean, Registrar and Vice chancellor to verify and approve the request. Above application should be implemented using spring framework with MVC technology with suitable front end techniques and Database servers..
Use the technologies: HTML5, CSS5, Spring framework, MySQL/Oracle.

Use Case-4: Stock management with CRUD operations using hibernate:

A Pharma Agency decided to manage their business using software. Monitoring Stock entry, sales, shortage, offers, profit and expired medicines are the requirements from the client using the targeted software. Since inserting details, update the offer details and removing expired stock entries, CRUD supported implementation expected to build. Instead of using JDBC API, hibernate suggested to be used for making efficient CRUD operations with database servers.
Use the technologies: HTML5, CSS5, Hibernate framework, MySQL/Oracle.

Total: 30 Hours

H. Learning Resources

i. Text Books:

1. Herbert Schildt, “Java: the Complete Reference”, Eleventh edition, Tata Mc-Graw Hill, 2019. [Unit 1, 2, 3]
2. Jim Keogh, “J2EE: The complete Reference”, First Edition, Tata Mc-Graw Hill, 2017. [Unit 4]
3. K. Santosh Kumar, “Spring and Hibernate”, McGraw Hill Education, 2017. [Unit 4,5]

ii. Reference Books:

1. H.M. Deitel and P.J. Deitel, “Java How to Program”, Pearson Prentice Hall Seventh edition, 2018.
2. Balaguruswamy, “Programming in java”, Sixth Edition, Tata McGraw Hill, 2019.
3. Jason Hunter, William Crawford, “Java Servlet Programming”, Second Edition, O'Reilly-2018.

iii. Online References:

1. “Programming in Java”, Accessed on: Apr. 20, 2021 [Online], Available: https://nptel.ac.in/courses/106/105/106105191/
2. “Java Tutorial”, Accessed on: Apr. 20, 2021 [Online], Available: https://www.javatpoint.com/java-tutorial

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC207	Database Management Systems	3	0	2	4

A.Preamble

Databases form the backbone of all major applications today – tightly or loosely coupled, intranet or internet based, financial, social, administrative, and so on. Structured Database Management Systems (DBMS) based on relational and other models have long formed the basis for such databases. While DBMS differ in the details, they share a common set of models, design paradigms via a Structured Query Language (SQL) or NoSQL. More specifically, this course introduces relational data models; entity-relationship modeling, SQL, data normalization, database design and modern databases asd well lightweight databases.

B.Prerequisite Course

10210CS101 – Problem Solving using C

C.Course Objectives

Learners are exposed to

- Design a database at basic level like Entity-Relationship modeling, Relational Modeling.
- Devise a database using Armstrong axioms, functional dependencies and views.
- Acquire knowledge of Query processing and concurrency transaction control.
- Learn an implementation issues with nontraditional databases.

D.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Apply basic database designing methodologies.	**K3**
CO2	Design a database schema for a given problem.	**K3**
CO3	Apply normalization for the given database.	**K3**
CO4	Use the properties of transaction and recovery management.	**K2**
CO5	Implement CRUD operations in modern database.	**K3**
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of Cos with Program Outcomes and Programme Specific Outcomes

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	2	2	2									1	
CO2		3	3	2	3							1	2
CO3		3	3		2							1	
CO4		2	2										2
CO5		2		2	3							1	2

High- 3; Medium-2; Low-1

F.Course Contents

Unit 1 Introduction to DBMS L–9 Hours

Purpose of Database System – Database Schema and Instances- Views of data – Database Languages - Database System Architecture – Database users and Administrator – Distributed Databases-DDB System Architecture-Database models-Entity–Relationship model – E-R Diagrams - Introduction to relational databases –Structure of relational databases- Relational model-Basics- From E/R diagrams to Database Design-Hierarchical model-Tree-structure diagrams- data retrieval-update facility-virtual records-mapping of hierarchies to files-Network Model- Data-structure diagrams-DBTG model-data retrieval-Find-Get-Set processing-update facility-Mapping network to files. Case study: Conference Management system.

Unit 2 Database Design and Query Optimization L–9 Hours

Keys and Integrity Constraints - Relational Algebra – Relational Calculus-Tuple – Structured Query language(SQL) Basic and additional Operations –Database Languages Revisited –create-alter- update-insert-rename-truncate-drop-select query commands-grant-revoke-rollback-commit control commands-Operators-aggregate functions-string-indent-date functions-set-union-union all-aliasing-Nested Queries & Join Queries–Join-Natural-inner-equi-cross/cartesian-outer-left-right-full outer join-Common Table Expressions-limit function in statistics-summarizing-Embedded SQL- Indexes and Triggers –compound- Views Definitions -Materialized view-Assertions-Query Processing and Optimization-Heuristics and cost estimates in Query Optimization-Tuning. Case study: Restaurant chain for network model.

Unit 3 Functional Dependencies and Normalization L–9 Hours

Introduction and problem of data redundancy-Features of good Relational database design- Functional Dependencies - Normalization – First Normal Form-minimal cover-canonical cover-Decomposition- Second Normal Form- Third Normal Form – controversies with SQL in normalization-Advanced Normalization -Boyce/Codd Normal Form, Fourth Normal Form and Fifth Normal Form- Dependencies preservation-Denormalization-Case Studies of database system like Air-cargo services**.**

Unit 4 Transaction and Concurrency L–9 Hours

Transaction Concepts –Schedules- Transaction States - Concurrent Execution-Serializability- -Two Phase locking Protocol –Timestamp ordering Protocol-Validation Based Protocol-Multiple Granularity-Intention lock modes-Multiple granularity locking scheme-Replication in Distributed databases-Deadlock-Handling-Prevention-Wait-diescheme- wound-wait scheme- Detection-Wait-for graph-Phantom phenomenon-Distributed deadlock detection-Index locking-weak levels of concurrency- Concurrency in index structures- B+ Tree crabbing- -Types of Failure-Recoverability -Recovery and

Prevention- Log based recovery-undo-redo-checkpoint-Recovery Algorithm-Transaction rollback with Logical undo- Recovery Algorithm with Logical Undo-ARIES Recovery Algorithm. Case study: Instances and Log recovery process in Oracle and Microsoft Access.

Unit 5 Storage and Modern Databases L–9 Hours

RAID storage -Modern databases-XML Databases- CAP Theorem-Datalog-Basic structure-Syntax-semantics-Relational operation in Datalog-Recursion in Datalog-Power of Recursion-Monotonicity-Non-monotonicity-NoSQL-Data models-Key value store, column families-Document Databases-MongoDB-Introduction to MongoDB- CRUD operations-Create-Insert-Update-Query-Removing documents-Sharding- Introduction to Mongoose-core concepts-extending models-CRUD operations with mongoose. Case study: Designing local library in Amazon using mongoose-SQLite-Building and Installing SQLite-DDL-DML-TCL-Select-compound select- alternate Join- JOIN..ON, JOIN..USING-Database design-SQL functions and SQLite extensions-Virtual Tables and Modules.

Total: 45 Hours (L)

G.Laboratory Experiments

PART - 1

Task 1: Conceptual Design after FTR (Tool: Padlet/ Creately)

Using basic database design methodology and ER modeler, design Entity Relationship Diagram by satisfying the following sub tasks:

1. a Identifying the entities.
1. b Identifying the attributes.
1. c Identification of relationships, cardinality, type of relationship.
1. d Reframing the relations with keys and constraints.

Task 2: Generating design of other traditional database model

Creating Hierarchical /Network model of the database by enhancing the sound abstract data by performing following tasks using forms of inheritance:

2. a Identify the specificity of each relationship, find and form surplus relations.
2.b Check is-a hierarchy/ has-a hierarchy and performs generalization and/or specialization relationship.
2. c Find the domain of the attribute and perform a check constraint to the applicable.
2. d Rename the relations.
2. e Perform SQL Relations using DDL, DCL commands.

Task 3: Using Clauses, Operators and Functions in queries

(Tool: GU/ Table Normalization Tool, ALM/PL-Padlet)

Perform the query processing on databases for different retrieval results of queries using DML, DRL operations using aggregate, date, string, indent functions, set clauses and operators.

Task 4: Using Functions in queries and writing subqueries:

Perform the advanced query processing and test its heuristics using the designing of optimal correlated and nested subqueries such as finding summary statistics.

Task 5: Writing Join Queries, equivalent, and/or recursive queries
Perform the advanced query processing and test its heuristics using the designing of optimizing complex queries and their equivalence queries

Task 6: Procedures, Function and Loops:
Programming using PL/SQL Procedures, Functions and loops on Number theory and business scenarios like..

Task 7: **Triggers, Views and Exceptions**
Conduct events, views, exceptions on CRUD operations for restricting phenomenon.

Task 8: **CRUD operations in Document databases**
Perform Mongoose using NPM design on MongoDB designing document database and performing CRUD operations like creating, inserting, querying, finding, removing operations.

Task 9: CRUD operations in Graph databases
Perform GraphQL/Neo4jgraph space design for recommendation engines. Also perform CRUD operations like creating, inserting, querying, finding, deleting operations on graph spaces.

Task 10: Normalizing databases using functional dependencies uptoBCNF
Upon relational tables created in task-2, perform normalization up to BCNFbased on given Dependencies as following for the assumed relations specified below.

10. a Apply the functional dependency, normalize to 1NF
10. b Normalize the relations using FD+ and α+
10. c Find the minimal cover, canonical cover.
10. d Normalize to 2NF, add/alter constraints if necessary.
10. e Normalize to BCNF, add/alter constraints if necessary.
10. f Normalize to 3NF, add/alter constraints if necessary.

Task 11: Menus, Forms and Reports:
For an application, creating and debugging Menus, Forms and reports using Oracle Forms and Report Builder or SQL server management studio or SQLite or NetBeans

TASK 12: Micro Project:
Develop microproject based on business scenarios and use cases like.

Total: 30 Hours

PART - 2

Use Cases

Use Case-1: Building a Cart analysis for Myph

Myph has just launched their brand new phone range to the eager reception of the consumer market cart analysis. The product's data model has a unique menu that identifies the product, title, description, a stock quantity, and pricing information about the item. All products have categories. To be able to provide a list of all the products in a category, amend the data model with a collection of documents for each category and contain the path for that category in the category tree. Use cart analysis in developing different consumer selection options. Would this answer outlier selection in cart i.e., surplus selections? Is Relational database application can answer these transactions? How recovery is made through carting and commerce?

Use Case-2: Indexing various devices in IoT platform

A generic IoT platform required support for data from a wide range of devices, some of which could not be envisaged while developing the platform. The proficient work necessitated a data storage mechanism that could handle data from different types of devices. Indexing support makes it easy to pull data using a single index or multiple indexes such as device id with location id. Records for a particular device in different locations are easily accessed. Common parameters like temperature from different types of devices and their records are retrieved fast through these indexes. How the application could lead to the choice of JSON-based document database, MongoDB? Assume or create JSON script in support of this.

Use Case-3:A gift coupon application that handles offers and payment-related information.

Choose the database system, a relational database, for its capability to scale horizontally while keeping the ACID property. The last was particularly important because transactional data was being handled. Eventual consistency as offered by other databases was not suitable. Going with DB and cluster gives the opportunity to scale horizontally for a large number ofwrites and reads without compromising the ACID and transactional capability required by the application. Will it transact and lead to no deadlock, keeping all relational tables normalized? If so till what normal form do they sustain to offer a gift coupon application?

Use case- 4: Army Supply chain, Bill of Materials and Maintenance Cost Management

The nation's armed forces, support more than one million soldiers and about 200,000 civilian staff. Each of these staff members relies on multiple pieces of equipment, from helicopters and armoured vehicles to small arms and radios, to complete their missions. With maintenance, operation and support costs of equipment representing as much as 80% of total lifecycle costs, it's imperative that the Defence ministry track and analyse equipment maintenance costs including changing historical data sources Dimension with more flexibility like graph databases and to be given richer analysis like forecast replacement parts with the location and climate, mean time to failure rates, logistics and requirement processes. Is answerable the vital "what-if" questions such as the cost of deploying certain forces and supporting equipment to a new war zone? Could the model perform multi-dimensional cost comparison and trend analysis? Will the solution promises how the data management in unpredictable maintenance costs?

Total: 30 Hours

H. Learning Resources

i. Text Books:

1. Abraham Silberschatz, Henry F. Korth and S. Sudharshan, "Database System Concepts", Seventh Edition, Tata McGraw Hill, 2019.[Unit 1 - 5]
2. Shannon Bradshaw, E Brazil, Kristina Chodorow, "MongoDB: The Definitive Guide - Powerful and Scalable Data Storage", Third Edition, Shroff/O'Reilly Inc., January 2020.[Unit-5]
3. AgusKurniawin, "Python and SQLite Development", First Edition, PE Press, January 2021. [Unit-5]

ii. Reference Books:

1. Raghu Ramakrishnan et al, "Database Management Systems", Third Edition, McGraw Hill, 2014.
2. Elmasri Ramez, Navathe S, "Fundamentals of Database System", Seventh Edition, Pearson, 2017.
3. J.D.Ullmann et al, "Database Systems: The Complete Book", Second Edition, Pearson Ed, Inc, 2009.

iii. Online References:

1. "Database Management System Part-1&11" Mar.25, 2019. Accessed on Feb.20, 2021[Online].Available:https://infytq.onwingspan.com/en/viewer/web_module/lex_auth_01305427365738086410\06_shared?collectionId=lex_auth_01275806667282022456_shared&collectionType=Course.
2. "Database Design", Dec 31, 2009. Accessed on May, 05, 2021[Online]. Available: https://nptel.ac.in/courses/106/106/106106093/.
3. "Designing local library models", Accessed on: May 05, 2021[online]. Available:https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/mongoose.

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC208	Software Engineering	2	0	2	3

A.Preamble

Software Engineering is an engineering branch that relates to the evolution of software product using well-defined scientific principles, techniques, and procedures. This course is being offered in order to train the students to presents the concepts, methods and techniques necessary to efficiently capture software requirements, to plan the project with appropriate estimations, and to bring it to completion.

B.PrerequisiteCourse

10211CC204–Programming using Java / 10210CS101 - Problem Solving using C

C.CourseObjectives

Learners are exposed to

- Understand the best practices in software engineering.
- Develop the necessary skills to handle software projects in a principled way.
- Provide an in depth knowledge on the different phases of the software life cycle.
- Perform requirements analysis, estimation design and testing for applications.

D.CourseOutcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Identify the suitable process model for software development.	K3
CO2	Prepare the SRS document with project estimation and plan.	K3
CO3	Design the UML Diagrams to develop a software application.	K3
CO4	Apply the software design heuristics for quality improvement.	K3
CO5	Build Test cases using various testing strategies.	K3
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of COs with Program outcomes and Programme SpecificOutcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	2	1								2	
CO2	3	2	2	1	1							2	1
CO3	3	2	3	1	2							2	1
CO4	3	2	2	1	2							2	1
CO5	3	2	2	1	1							2	1

3-High; 2-Medium; 1-Low

F.CourseContents

Unit 1 Software Development Process Models L–6 Hours

Introduction to Software Engineering - Software Development process models – Waterfall Model, Incremental Model, Spiral Model, V Model – Agile Development- Agile Methodologies- Agile Principles- SCRUM and XP - Project management- Stages of Project management- 4 P’s of project management - Process & Project metrics

Case Study: The Railways tracking - Arrival Time Prediction System

Unit 2 System Requirements Specification, Planning & Scheduling L–6 Hours

Software Requirements Specification - Software project planning - Software Estimation - Empirical Estimation Models – Software Project Scheduling- Risk Management - Software risks – Risk identification – Risk projection – Risk refinement - Defect Prevention.

Case Study: Preparation of SRS for the camera motion sensor system for home automation.

Unit 3 Analysis using UML L–6 Hours

Analysis Modeling - Data Modeling - Functional Modeling& Information Flow - BehavioralModeling-Structured Analysis - Object Oriented Analysis - Domain Analysis-Object Relationship Model - Object Behavior Model, Design modelling with UML- Design modelling with UML.

Case Study: UML diagrams for Stock Trading Application using the online Creatly platform.

Unit 4 Design Principles and Heuristics L–6 Hours

Design Principles - Design Process - Design Elements- Design Concepts - Modular Design-top-down and bottom-up, Abstraction, step-wise refinement, coupling and cohesion, Refactoring, Architecture- Introduction to Software Architecture - Data Design - Transform Mapping - Transaction Mapping - Design Patterns.
Case Study: Architecture diagram and design for the Safe Home security system.

Unit 5 Testing, Maintenance and Reengineering L–6 Hours

Testing approaches -Top-Down, Bottom-Up, Big Bang and Sandwich, object oriented product Implementation & Integration- Software Testing methods- White Box- Black Box - Unit Testing - Integration testing - Validation & System testing – Software Maintenance & Reengineering.
Case study: Login module of Ecommerce application - Test cases and Best Practices.

Total: 30 Hours (L)

G.Laboratory Experiments

PART - 1

Task 1: Design of problem statement, perform a detailed feasibility study and finalize the process model to be used.

Tools: Microsoft Word

Task 2: To Prepare the Software Requirement Specification Document in the given template to include the functional and Non-functional requirements.

Tools: Perforce Helix RM and Microsoft Word.

Task 3: To Prepare a detailed estimate using the FP and COCOMO Model. Also prepare a detailed schedule of the project.

Tools: GanttPRO or Lucid Chart and Microsoft Word.

Task4: Identify Use Cases and develop the Use Case model for the given scenario Draw the relevant Entity Relationship Diagram and class diagram.

Tools: Creately, Rational Rose or any CASE Tools

Task5: Using the identified scenarios, find the interaction between objects and represent them using UML Sequence & Communication diagrams.

Tools: Creately, Rational Rose or any CASE Tools

Task6: To implement behavioural modelling using User Interface diagrams, state transition diagram and activity diagrams for the given scenario.

Tools: Creately, Rational Rose or any CASE Tools

Task7: To implement partial layered, Object Oriented and logical software architecture diagram to visualize the high level design.

Tools: Rational Rose or any CASE Tools

Task8: To Implement the data Flow Diagrams to perform the data modelling with Level 0, Level 1 and Level 2 analysis.

Tools: Creately, Rational Rose or any CASE Tools

Task9: To Develop test cases for various Testing methodologies. Write the test cases for both Black box Testing and White Box Test Testing.

Tools: TestopiaTools and Microsoft Word.

Task10: To Develop a prototype of the project with database management system and deploy the project.

Tools: Rational Rose or any CASE Tools

PART-2

Use Cases:

Use Case - 1:

Analyze and implement the Employee management and Payroll Processing application.

Use Case - 2:

Build the Library Management system application with project plans, schedule, estimation and design

Use Case - 3:

Design ,Analyze and implement the Passport Automation System application

Use Case - 4:

Identify, plan, analyze and implement the E Voting System application.

Total: 30 Hours

H. Learning Resources

i. Text Books:

1. Roger. S. Pressman and Bruce R. Maxim, “Software Engineering – A Practitioner’s Approach”, 9th Edition, McGraw Hill, 2020. [Unit 1, 2, 4, 5]
2. Ian Somerville, “Software Engineering”, 10th Edition, Pearson Education, New Delhi, 2017. [Unit 3]

ii. ReferenceBooks:

1. Fairley R, “Software Engineering Concepts”, Indian edition, Tata McGraw Hill, New Delhi, 2017.

iii. OnlineReferences:

1. Software Development Lifecycle, Accessed on: Apr. 20, 2021 [Online], Available:https://www.coursera.org/specializations/software-development-lifecycle
2. Writing Test Cases, Accessed on: Apr. 20, 2021 [Online], Available: https://www.softwaretestinggenius.com/a-simplistic-approach-to-writing-test-cases-the-black-box-way/
3. Introduction to Software Engineering, Accessed on: Apr. 20, 2021 [Online],Available: https://www.coursera.org/learn/introduction-software-testing

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC210	Big Data Analytics	3	0	2	4

A.Preamble

Thekey objective of this course is to make the students to be familiar with the concepts of data warehouses, modelling and design of data warehouse. This course includes the most significant technologies used for manipulating, storing, and analyzing big data. In addition, the basic tools of data warehouse and big data analytics will be explored. These tools make the students to understand the strategies involved in data warehouse design and Big Data Platform.

B.Prerequisite Course

10211CC207 - Database Management Systems

C.Course Objectives

Learners are exposed to

- Understand the principles of data warehousing, multidimensional data model.
- Familiar with the data warehouse architecture and OLAP tools.
- Impart the architectural concepts of big data file management system.
- Explore PIG and HIVE tools to develop applications in Big data analytics.
- Implement best practices for Big Data Optimization.

D.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Build and process the data warehouse modelfor given application.	K3
CO2	Design Job Execution procedures in Map Reduce and Apache SparkParadigm.	K3
CO3	Design memory efficient solutions for Big Data Applications Procedures.	K3
CO4	Build Big Data Solutions for social media network applications.	K3
CO5	Apply cloud and optimization techniques to develop the solutions for real time scenarios.	K3
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of COs with Program outcomes and Program Specific Outcomes

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3		3	3									3
CO2		2	2	2	3								3
CO3	2		3		3	2						2	2
CO4	2	3		3	3	3						3	3
CO5	2	2		3		3						3	3

3- High; 2-Medium; 1-Low

F.Course Contents

Unit 1 Data Warehousing and Business Analysis L-9 Hours

Introduction to Data warehousing – Evolution of Decision Support systems – Modeling a Data Warehouse – Granularity in the Data Warehouse - Data Warehousing Components, Building a Data Warehouse, Warehouse Database- Extract, Transform and Load: ETL Overview, ETL Requirements and Steps, Data Transformation, Data Loading, ETL Tools. –. Reporting and Query tools and Applications –Online Analytical Processing (OLAP) – Need – Multidimensional Data Model.

Unit 2 Big Data Processing L-9 Hours

Introduction to Big Data, Big Data Analytics, Evolution of Big data – Best Practices for Big data Analytics – Big data characteristics- Understanding Big Data Storage – A General Overview of High-Performance Architecture – HDFS – Map Reduce Programming Model-Understanding the basics of MapReduce, Loading data into HDFS, Introduction-Apache Spark, Features, Components, Resilient Distributed Datasets, Data Sharing using Spark RDD, Spark Programming.

Unit 3 Big Data Storage System L-9 Hours

Apache Camel- Introduction, Overview, features, camel context, architecture, message queues. Apache Ignite-Memory Architecture, Memory Pages, Lifecycle, In-Memory Data Grid, Caching Support, Streaming Support. Cassandra-Architecture, Data Model, Referenced Api, Cqlsh, Keyspace Operations, Table Operations, Batch, CRUD Operations. Apache Kafka- Fundamentals, Cluster Architecture, Work Flow, Basic Operations, Producer and Consumer Example.

Unit 4 Big Data Visualization and Prediction L-9 Hours

Pig : Introduction to PIG, Execution Modes of Pig, Comparison of Pig with Databases, Grunt, Pig Latin, User Defined Functions, Data Processing operators. Hive : Hive Shell, Hive Services, Hive Metastore, Comparison with Traditional Databases, HiveQL, Tables, Querying Data and User Defined Functions, NoSQL Databases : Schema-less Models‖: Increasing Flexibility for Data Manipulation-Key Value Stores- Document Stores – Tabular Stores – Object Data Stores – Graph Databases Hive – Sharding- Hbase – Analyzing big data with twitter – Big data for E-Commerce Big data for blogs.

Unit 5 Big Data Cloud Concepts and Optimization L-9 Hours

Big data Cloud Computing-Features, Cloud Deployment Models, Cloud Delivery Models, Cloud for Big Data,Real time Analytics Platform(RTAP) applications – Using Graph Analytics for Big Data: Graph Analytics, Big Data Optimization- Smooth Convex Optimization-Projection-free methods, Accelerated gradient descent methods, Non smooth Convex Optimization-Smoothing techniques, Mirror-Prox method, Sparsity learning,Large-scale kernel machines.

Total: 45 Hours

G.Laboratory Experiments

PART- 1

Task 1: Design an multi-dimensional data model schema namely Star, Snowflake and Fact Constellations for a Categorical data using SQL Server Management Studio (SSMS).
(Perform the above for Banking, Healthcare, Manufacturing, Sales and Automobile)
Tools: SQL Server Management Studio (SSMS), Microsoft Azure SQL Pool

Task 2: To configure, monitor, and administer a Data warehouse and perform basic Query operations on the DW.
Tools: SQL Server Management Studio (SSMS), Microsoft Azure SQL Pool

Task 3: Perform Data Cube Operations (OLAP Operations) using SQL Queries
Rollup
Rolldown
Slicing
Dicing
Database: MySQL

Task 4: Implement matrix multiplication with Map Reduce.
Tools:LINUX

Task 5: Write a Spark application to perform word count in the input file.
Tools: APACHE SPARK

Task 6: Implement CRUD operations on Casandra
Tools: Casandra

Task 7: Implementing Producer and Consumer problem in kafka.
Tools: kafka

Task 8:
i. Implement basis commands in HIVE.
ii. Use Hive to create, alter, and drop databases, tables, views, functions, and indexes **Tools: HIVE , LINUX**

Task 9: Write Pig Latin scripts sort, group, join, project, and filter the data
Tools: Pig, LINUX

Task 10:
i.Construct the Pig Latin Scripts to find character Count
ii.Construct the Pig Latin Scripts to find a max temp for each and every year.
Tools: Pig, LINUX

Task 11: Collect any Social Media Data from a Twitter to a Local File with the Topic ‘covid 19’. Download and Set Up MongoDB Server and a Client Mongo shell.
Tools: MongoDB, Python: Scipy

Task 12: Retrieve Analytic Information given below from MongoDB created in task 9:
i. For each “place_type”, Find total favorite_count
ii. For each "country_code", find total "retweet_count"
iii. Find out top 10 most frequent topic words of the entire tweet message texts of your collection after lemmatization/stemming and removing all the Stop Words.

Tools: MongoDB, Python: Scipy

PART - 2

Use Cases :

Use Case – 1:

Data analytics using Apache Spark on Amazon food dataset, find all the pairs of items frequently reviewed together.

Write a single Spark application that:

a. Transposes the original Amazon food dataset, obtaining a PairRDD of the type: user-id – list of the product-ids reviewed by user-id
b. Counts the frequencies of all the pairs of products reviewed together;
c. Writes on the output folder all the pairs of products that appear more than once and their frequencies.
d. The pairs of products must be sorted by frequency.

Use Case – 2:

Construct MapReduce program to perform data analysis on weather dataset.

Dataset is available at: https://www.kaggle.com/datasets/zaraavagyan/weathercsv

Use Case – 3:Apache Hive for Rea in real-world applications

It is often the case that data has to be retrieved in real-time from the source. The processing is done as soon as the data is inputted. For example, Google maps processes traffic data in real-time. As soon as it receives information from the source, the data is output onto its application. Develop the Real-time data processing using Apache Hive for analysing the Google Map data.

Use Case – 4:Build a Data Pipeline based on Messaging using PySpark and Hive

Implement a data pipeline that ingests raw data from a source and moves this data to a destination where it can be stored or processed further for analysis. The pipeline should also able to filter or clean the data for various purposes. Then create a Hive external on top of HDFS which will allow the cleaned, processed data to be deployed.

Total: 30 Hours

H. Learning Resources

i. Text Books:

1. Jiawei Han and Micheline Kamber, “Data Mining Concepts and Techniques”, Third Edition, Elsevier, 2012. [Unit 1]
2. Seema Acharya, Subhasini Chellappan, "Big Data Analytics" Wiley 2015. [Unit 2-5]

ii. Reference Books:

1. W.H. Inmon, “Building the Data Warehouse”, John Wiley & Sons, Inc, 4th Edition, 2005-(Online Publication: 2014).
2. Capriolo, E., Wampler, D., & Rutherglen, J., “Programming Hive”, O’Reilly Media, Inc.”,2012.

iii. Online References:

1. “Data Warehouse Design: Modern Principles and Methodologies”, Accessed on Oct 27, 2022 [online]. Available: https://cdn.ttgtmedia.com/searchDataManagement/downloads/Data_Warehouse_Design.pdf
2. “Big data Analytics”, Accessed on Apr. 5, 2022 [online]. Available: https://www.analyticsvidhya.com/learning-paths-data-science-business-analytics-business-intelligence-big-data/learning-path-data-science-python/
3. “Apache Spark Tutorial”, Accessed on Oct 28, 2022 [online]. Available: https://www.tutorialspoint.com/apache_spark/apache_spark_deployment.htm
4. “Apache Camel Tutorial”, Accessed on Oct 28, 2022 [online]. Available: https://www.tutorialspoint.com/apache_camel/apache_camel_quick_guide.htm

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC212	Web and Mobile Application Development	3	0	2	4

A.Preamble

This course provides basic concepts about HTML5, CSS3, Bootstrap Framework and java script technologies to create the interactive client-side design of web applications. This course also teaches the server-side programming using Node JS, PHP and MVC frontend design for web applications using Angular. The conversion of web application to mobile apps can be performed using Angular. Finally, the creation and deployment of micro services using Seneca and Dockers will be discussed.

B.Prerequisite Course

10211CC204- Programming Using Java

C.Course Objectives:

Learners are exposed to

- Design the interactive and responsive web application.
- Build the efficient server-side applications.
- Develop the single page MVC application.
- Construct and deploy the Microservices.
- Conversion of web apps to Mobile apps using Angular.

D.Course Outcomes:

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Design web applications using Hypertext Mark-up Language and Cascading style sheets through bootstrap.	**K3**
CO2	Build the interactive and dynamic web page using Java Script Technology.	**K3**
CO3	Implement the server-side business logic to handle client request using NodeJS and PHP.	**K3**
CO4	Make use of MVC framework for integrating the window controls and its corresponding actions through event handlers.	**K3**
CO5	Transform the function as Microservice components to enhance the Reusability concept.	**K3**
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of COs with Program outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1		2	3		3	1						3	
CO2		3	3	1	2	1						3	
CO3		3	2		3	2						2	
CO4		3	3		3	2						3	
CO5		3	3		3	2						3	

High- 3; Medium-2; Low-1

F.Course contents

Unit 1 Front end design using HTML5, CSS3 and BOOTSTRAP Framework L-9 Hours

Introduction to HTML5: Basic Elements, Form Elements, Media Elements, HTML5 Graphics (Canvas, SVG), **CSS3:** Selector String, Box Model, Text Properties, CSS 3D Transformation, CSS Animation, **Bootstrap Framework:** BS Grid, Tables, Images, Alerts, Form Elements. Case study: Online Blog Creation.

Unit 2 Dynamic web page design using Java Script and jQuery L-9 Hours

Java Script: Data Types and Variables - Operators - Control Statements - Functions -Objects - Build in Objects - DOM - Java Script Event Handling - Form Handling and validations - **AJAX &JQuery:** Introduction to AJAX and JQuery: Introduction - JQuery Selectors - JQuery Animations - Effects - Event Handling - JQuery DOM Traversing-JSON - JQuery AJAX. Case Study: Web Scraping and Automation.

Unit3 Server-side programming using PHP and Node JS L-9 Hours

PHP: Introduction - Variables - Program control - Builtin Functions - Connecting to Database using MySQLi - Cookies - Sessions - Regular Expression; **Node JS:** Introduction - Architecture - Features of Node JS - Installation and setup - Function - Module - Creating Web Servers with HTTP (Request & Response) - Event Handling - Express JS - Get and Post Implementations - Database connectivity. Case Study: online Book Store.

Unit4 Single Web Page Design using Angular L-9 Hours

TypeScript: Introduction to Type Script - Variables - Datatypes - Enum - Array - Tuples - Functions - OOP concepts - Interfaces - Generics - Modules - Namespaces - Decorators. **Angular:** components and Modules - Data Binding - Event Binding - Templates - Directives - Services - Dependency Injection - Routing and Navigation pages - Template based Form - Reactive Forms - Validating - Pipes - Sorting and Filtering -Decorator - HTTP Client - Data storage - Observables &RxJS Case study: Todo list.

Unit5 Creating Forms and Mobile app Conversions L-9 Hours

Microservices: Introduction - Microservices Architecture - Microservices in Node.js: Installing Node.js, NPM, Seneca and PM2-SOLID Design Principles - Seneca Toolkit: Inversion of control(IOC) - Pattern Matching - Reusing Patterns - Writing Plugins - Web Server Integration - Data Storage - PM2 Task Runner for Node.js - Writing Microservices - Integrating with Express - Testing and Documenting Microservices - Monitoring Microservices using PM2 and Keymetrics - Deploying Microservices. **Converting Web App to Mobile App Development**- Install Capacitor Package - Configuration Setup - Native IOS and Android packages - Build and Deploy the application. Case study:Chat Bot.

Total: 45 Hours

G.Laboratory Experiments

PART – 1

Task1: Develop a Simple College Website including all the Department Information using HTML5 and CSS3.

Task 2: Create Home Page, Sign up and Login Page for Clinic Management Service using Bootstrap Framework.

Task 3: Validate the Registration, user login, user profile and payment by credit card pages using JavaScript.

Task 4: Parse the web page to get the required information using JQuery and DOM Traversing.

Task 5: Create a simple HTTP web server using Node.js to generate a dynamic response

Task 6: Create a three-tier application using Node.js and MySQL data base.

Task 7: Create an Reactive form for User Registration using Angular for Online Exam portal.

Task 8: Develop web application to implement routing and navigation in Angular.

Task 9: Develop a micro service for finding what people think by asking 500 people's opinion for any consumer product in Node.js using Seneca Toolkit.

Task 10: Build the Web application and convert it into Mobile app.

PART-2

Use Cases:

Use Case – 1: Bike Rental System

This system is named as **Bike on Rent Management System**. This system is designed to help the customers to take bikes or two-wheelers on rent. When we go on any trip outside the town or country we want to be free of time so instead of going through metros and taxis we prefer to have our own vehicle for rent. Using this system vehicle owner can register assellers and customers who want to take bikes on rent can register them as renters and can take any bike on rent. Address of the both are required as the customer can only take bike by going to the address provided and the vehicle owners can know the address that a customer is verified or not. The customer also has to upload some proofs to take the bike on rent. Proofs like license, pan card and identity card are compulsory so that no one could run taking the bike. Any customer whose proofs are not uploaded and are not valid will not be allowed to take any bike on rent. This has one admin account who verifies the registering user and two types of the user account. One for bike sellers and one for customers who take the bike on rent. This system has only one admin account and cannot have more than one admin account. There will be simple chat room needs to be added to make instant interactions between customer and admin. Admin can verify and register the user who is registering. If the admin does not verify, the user cannot register.

Tools:HTML5, CSS3, BOOTSTRAP, Angular, NodeJS, JQuery

Use Case – 2: Clinic Management Service

This system is named as Clinic management service. This system is made to keep the records about the patients, doctors and other staff members working at a clinic or hospital receptionist. One can login into the clinic management service using the email id or the user id and password. After signing in into this system there are the options to add new patients, new doctors and other new staff members like nurses and ward boys etc. This system is designed to easily maintain the data of the patients specifically. Daily many new patients visit the clinic so adding the new patient's details and keeping the records using it is very easy. There is also an option to add and delete doctors and other staff member's details. New and unique ids are given to everyone who gets registered over this system. There are also the options to check the patient's disease and course the patient is going through. Fees paid by any customer or patient can be saved on it and it is easy to calculate daily that the money is collected. Doctors and another staff member like nurses, ward boys, janitor and maid leaves can be deducted from their salary and rest of the salary can be paid easily using this clinic management service. It is easy to calculate money and handle account on monthly basis also. This system also has an option to use the backup that means if we are backing up the database it can never be lost. So, overall, this clinic management service is a solution to all the problems that we face in a clinic or hospital.

Tools: HTML5, CSS3, BOOTSTRAP, Angular, NodeJS, Java Script

Use Case – 3: E-commerce service

Online shopping is a process in which people (specifically customers) are being provided with the option of purchasing goods and services directly from the seller, all in real-time environment. Online shopping is an application of the internet as electronic commerce (Like Amazon). From the business perspective, customers usually find the products more attractive, on websites, as they get all the details available there. People in large number are doing online shopping today, and it is not only because it is convenient as one can shop from home, but also because there are ample amount of varieties available, with high competition of prices, and also it is easy to navigate for searching regarding any particular item. For sellers, their products have access to World Wide market, which also increases the number of customers and enhances the customer relationships. Also the web stores are a means for the small scale companies to launch their products at global level. The main objective behind this project is to develop a web oriented application which can provide an online shopping feature to the users. In other words, the project aimed at creating a virtual shop environment for users, in some handy form, which will be available to them through internet. This system has been designed keeping in mind all the aspects such as loading the data, complexity and maintaining the security of user credentials.

Here in this system, complexity refers to the total number of features being provided to users, and their smooth arrangement and functioning required. Following are some of the key features of our system, which distinguishes it from others:

- Display of all the available categories for shopping on the home page.
- Display all the sub-categories on the home page; those are associated with any particular item.
- Admin has the authority to add new particulars to the items list whenever needed.
- Permission to administrator to remove items, anytime.
- Allows the admin to modify the price of each item, whenever required or felt like.
- Admin has the authority to update the description of each item.

- Permission to the admin to view information about each customer who checkouts the items list.

Tools: HTML5, CSS3, BOOTSTRAP, Angular, NodeJS, Java Script

Use Case – 4: College management service

This application college management system based on Internet that aims to all the levels of management providing information within an organization. This system can be used as an information management system for the college. For a given student/staff (Technical / Non-technical) the Administrator creates login id & password, using these student/ staff (Technical / Non-technical) can access the system to either upload or download some information from the database. Not only will this added user also get to know about the events and extra curriculum activities which will hold into the college campus.

The main menu will contain six parts which are as follow:

- Student Login
- Teacher login
- Deposit Fee
- Ask Queries Forum
- About Us
- Contact Us

Now, we can see that nearly everything is very much possible to perform with a single click, so this application will help you to computerize the system of college management so that student and other staff members can access the system online.

These are some key features of the system which is as follow:

- ✓ To reduce the headache of maintaining the record of students and teachers related documents.
- ✓ To reduce the cumbersome job of maintaining several documents like
- ✓ It will eliminate the delays in the generation of results and free updating of the students; this system will help in maintaining the records of absent students.
- ✓ Searching will become more efficient and fast in comparison of manual searching.
- ✓ It will also provide assurance that each employee of the college marked their attendance timely.
- ✓ Overall it will reduce the cost and time of the college head in taking care of the college.

Tools: HTML5, CSS3, BOOTSTRAP, Angular, NodeJS, Java Script

Use Case – 5: E-Payment

In all E-commerce and other online services, payment activity is the most important section and this section mostly having the common procedures. If we make the payment service as the micro service component any web application full stack developer can easily integrate with his project instead of redesigning which is extra burden to developer. So, create the component for the Payment activity with the following operations:

- ✓ Getting card details
- ✓ Pin verification

- ✓ Encryption of card details
- ✓ Crediting and debiting the money on respective accounts.
- ✓ Payment Acknowledgement.
- ✓ Invoice Generation.

Convert this application to Mobile app

Tools: Node.js, Angular

Total: 30 Hours

H. Learning Resources

i. Text Books:

1. Ben Frain“ResponsiveWeb DesignwithHTML5 and CSS:Developfuture-proofresponsive websites using the latest HTML5 and CSS techniques”, Packt PublishingLimited,3rd Edition, 2020. [Unit 1]
2. MaryDelamater,“Murach'sJavaScriptandjQuery(4thEdition)Paperback–Illustrated”,MikeMurach&AssociatesInc, August26, 2020. [Unit2]
3. DavidHerron“Node.jsWebDevelopment:Server-sidewebdevelopmentmadeeasywithNode14usingpracticalexamples”,PacktPublishingLimited,5thEdition,2020.[Unit3]
4. YakovFain“AngularDevelopmentwithTypeScript”,ManningPublications, 2ndEdition,December2018.[Unit 4]

ii. ReferenceBook:

1. AzatMardan,“PracticalNode.js:BuildingReal-WorldScalableWebApps”,1stEdition.2018.

iii. OnlineReferences:

1. “Angularthecompleteguide(2021Edition)”,Accessedon:May.6,2021[Online],Available:https://www.udemy.com/course/the-complete-guide-to angular-2/
2. “Server-sideDevelopmentwithNodeJS,ExpressandMongoDB”,Accessedon:May.06,2021 [Online],Available: https://www.coursera.org/learn/server-side-nodejs
3. “Node.jsMicroservicesfor beginners”,Accessedon:May.06,2021[Online],Available:https://www.udemy.com/course/nodejs-microservices-for-beginners/

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC219	Cyber Security	3	0	2	4

A. Preamble

Cyber Security is the body of technologies, processes, and practices designed to protect networks, computers, and data from attack, damage, and unauthorized access. This course provides the skills in cyber security in view of cybercrime, cyber offenses, frauds in mobile and wireless devices, handling techniques of cybercrime, organizational implications and cyber terrorism.

B. Prerequisite Course

10211CC130 - Fundamentals of Computer Networks

C. Course Objectives:

Learners are exposed to,

- Assess threat models and their influence on an organization
- Compare the various uses and approaches to cryptography
- Prepare for and respond to security incidences
- Design effective and efficient password schemes
- Plan environments that are resistance to malware

D. Course Outcomes: Upon the successful completion of the course, students will be able to:

CO No’s	Course Outcomes	K - Level
CO1	Understand the basic concepts of cyber security and its principles.	K2
CO2	Identify the attacks, security issues, and management standards in mobile and wireless devices.	K2
CO3	Apply the tools and methods to detect cyber attacks	K3
CO4	Identify the phishing techniques and associated vulnerabilities.	K3
CO5	Use the preventive measures for cyber security safeguards.	K3
Knowledge Level (Based on revised Bloom’s Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of COs with Program outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	2	2				2	2	1					
CO2	2	3		3		2	2	1				2	2
CO3	1	2		2	3	2	2	2				2	3
CO4	1	3			3	2	2	2				3	3
CO5	2		3	2	3	2	2	2				3	3

High- 3; Medium-2; Low-1

F.Course contents

Unit 1 Introduction L-9Hours

Introduction: Definition and Scope- Risks- Threats- Classifications of Cybercrimes- The Legal Perspectives: Indian Perspective- Global Perspectives. Cyber offenses: Categories of Cybercrime- Social Engineering and its classification- Cyberstalking- Cybercafe and Cybercrimes- Botnets.

Unit 2 Cybercrime: Mobile and Wireless Devices L-9 Hours

Trends in Mobility: Credit Card Frauds in Mobile and Wireless Computing Era, Security Challenges Posed by Mobile Devices, Authentication Service Security, Attacks on Mobile/Cell Phone - Identity and access management – Architecture – IAM Standards.

Unit 3 Tools and Methods used in Cybercrime L-9 Hours

Introduction: Proxy Servers and Anonymizers, Password Cracking: Online Attacks, Offline Attacks, - Control Mechanisms - Preventive and Detective controls - Metasploit framework - Keyloggers and Spywares: Software Keyloggers, Hardware Keyloggers, Antikeylogger, Spywares.

Unit 4 Phishing and Organizational Implications L-9 Hours

Phishing and Identity Theft: Phishing, Types of Phishing - Phishing Countermeasures - Virus- Worms- Trojan Horse-Identity Theft (ID Theft)- Cybersecurity vulnerabilities – vulnerabilities in software and system administration- Web threats for organizations.

Unit 5 Cyberterrorsim L-9 Hours

Introduction- Access control- Audit- Authentication- Biometrics- Cryptography- Deception- Denial of Service Filters- Ethical Hacking- Firewalls- IDS- IPS- Scanning- Security policy- Threat Management.

Total: 45 Hours

G.Laboratory Experiments

PART – 1

Task 1: Installation of VM work station and Kali Linux Operating System.

Task 2: Identify the vulnerabilities and exploit attack vectors on a target device by scanning different parameters to accomplish common platform enumeration and common vulnerabilities exposure using legion in kali linux.

Task 3: Retrieving PII of a host such as a registrant name, organization, country, name servers, and date of creation, expiry, and updation.

Task 4: Port Scanning using network mapper (nmap) in kali linux to expose the active and non-active ports and its associated vulnerabilities.

Task 5: Managing network traffic using wireshark tool.

Task 6: Apply steganography attack using OpenStego.

Task 7: Create and verify the virus file**.**

Task 8: Terminate and Stay Resident (TSR) program for identify the virus.

Task 9: Introduction to operations on Metasploit framework and Scripting for penetration testing – Recon and Enumeration scripts

Task 10: Configuring Windows and Linux firewall and managing user permissions

PART – 2

Use Cases:

Use Case - 1:

A National Vehicles Insurance company contacting you for assistance regarding a cyber incident raised by the company. The company customers' data leaked to the internet and some investigations indicate that their Customer Management System which runs on IP Address was targeted by somebody and had stolen critical data.Using your penetration testing skills & techniques, find the system weaknesses & investigate the incident.

Use Case - 2:

In a Denial-of-Service (DoS) attack, threat actors flood a targeted system’s network by directing lots of traffic towards it, from multiple systems under their control. It is a common tactic used by attackers via a network of compromised systems to render an online service unusable. DoS attacks can end up hurting an organization’s reputation by affecting its services uptime, customer activity, and business operations. The motives behind DoS attacks can also include extortion, hacktivism, cyber warfare, etc., Explain the process to Detect and respond to break DOS attacks.

Total: 30 Hours

H. Learning Resources

i. Text Books:

1. Nina Godbole, SumitBelapure, “Cyber Security”, Willey India, Edition 1, 2012.
2. B. B. Gupta, D. P. Agrawal, Haoxiang Wang, “Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives”, CRC Press, ISBN 9780815371335, 2018.
3. Raef Meeuwisse, “Cyber Security for Beginners”, Cyber Simplicity Ltd., 2017.

ii. References Books:

1. Gary M. Jackson, “Predicting Malicious Behavior: Tools and Techniques for Ensuring Global Security”, John Wiley & Sons Publisher, June 2012.
2. Roger Grimes, “Hacking the Hacker”, Wiley India, 2017.
3. Donaldson, S., Siegel, S., Williams, C.K., Aslam, A., “Enterprise Cybersecurity - How to Build a Successful Cyber defense Program against Advanced Threats”, A-press, 2015.

iii. Online References:

1. The Complete Cyber Security Course: Hackers Exposed”, 2018. [online]. Available: https://www.udemy.com/the-complete-internet-security-privacy-course-volume-1/, [Accessed: May 2022].
2. Cyber Security Pentesting Tools**,** [online]. Available: https://www.cybrary.it /0p3n/7-cyber-security-pentesting-tools/ , [Accessed: May 2022].
3. Cyber security insights, [online], Available: https://www.sans.org/reading-room/, [Accessed: May 2022].

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC224	Full Stack Application Development	1	0	4	3

A. Preamble

This course provides practical knowledge of database systems, web development, backend programming, and cloud technologies. Students gain hands-on experience in building dynamic web applications using modern frameworks, version control tools, microservices, DevOps practices, and cloud platforms. The course focuses on applying concepts to develop secure, scalable, and deployable software solutions.

B. Prerequisite Course - Nil

C. Course Objectives

Learners are exposed to:

- Understand and use database queries and web technologies to build interactive applications.
- Apply Spring Boot concepts to develop backend and RESTful services.
- Use microservices architecture for building scalable applications.
- Practice version control, build tools, and DevOps workflows for software deployment.
- Explore cloud platforms and AI-assisted coding techniques for modern application development.

D. Course Outcomes

Upon the successful completion of the course, students will be able to:

COs	Course Outcomes	K Level
CO1	Use database queries and web technologies to create dynamic web pages.	K3
CO2	Implement backend services using Spring Boot and RESTful APIs.	K3
CO3	Develop microservice-based applications using Spring frameworks.	K3
CO4	Apply version control, build tools, and CI/CD practices for application delivery.	K3
CO5	Build cloud-enabled applications using cloud services and AI-based coding tools.	K3
Knowledge Level (Based on revised Bloom’s Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of COs with Program Outcomes and Programme Specific Outcomes

Cos	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	2	2	3	1	2	-	-	-	-	-	-	2	3
CO2	3	2	3	2	3	-	-	-	-	-	-	2	3
CO3	3	3	3	2	3	-	-	-	-	-	-	2	3
CO4	2	2	2	2	3	-	-	-	-	-	-	1	3
CO5	2	2	3	2	3	-	-	-	-	-	-	1	3

3- High; 2-Medium; 1-Low

F. Course Contents

Unit I: Database Management and Web Development basics L - 3 Hours

SELECT statement – Aggregate functions - Sorting and Filtering Data - Joins - Subquery. Working with Date- Triggers, Indexes, and Views - Transactions, Commit, and Rollback – SQL Built-in and User-Defined Functions. HTML5 – Tags in HTML – Design HTML Forms, Handling user inputs, - use of required. CSS3. JavaScript for Dynamic Web - ES6 features - DOM manipulation – GetElementById - Form validation. Event handling - Mouse click and double click events - Keyboard events – Developer tools.

Unit II: Spring Boot Core and Backend Development L - 3 Hours

Spring MVC – Annotation - Spring Core: Inversion of Control and Dependency Injection – Auto wired - BeanFactory- Spring Boot Web Stack (Servlet-based Architecture) - Object-Relational Mapping Related Annotations - Data Access Layer using Spring Data JPA. RESTful API Design and Development - Exception Handling and Validation – Spring Security Implementation.

Unit III: Advanced Spring, Microservices Programming L – 3 Hours

Microservices Design Principles - Service Registry and Discovery - API Gateway and Load Balancing - Spring Messaging (JMS) – React JS - Web application into mobile application – Capacitor package.

Unit IV: Version Control, Build Tools and DevOps Foundations L - 3 Hours

Git & GitHub Fundamentals – Repository Management - Versioning - Branching, Merging. Build Automation using Maven - DevOps - Continuous Integration and Continuous Deployment (CI/CD) using Jenkins, GitHub Actions, and GitLab CI/CD.

Unit V: Cloud Computing and Vibe Coding L – 3 Hours

Cloud Services: API Gateway, File Storage, Compute, Relational Database Service– Pay as you Use model - Introduction to AWS - Google Cloud - Azure - Platform as a Service (PaaS) – Infrastructure as a Service (IaaS) - Software as a Service (SaaS) – Cloud applications for business. Vibe Coding - Prompt Engineering – Fine tuning the prompt - Generative AI.

TOTAL: 15 Hours

G. Laboratory Experiments

Part A

Task 1: Student Registration & Data Storage

Design a Student Registration Form using HTML5 and CSS3 that collects:

Name, Email, DOB, Department, Phone

Store the data in a database table and retrieve it using SELECT.

Task 2: Data Retrieval & Sorting Dashboard

Description: Create a page that displays student or employee records with:

Sorting by name or date

Filtering by department
Count of students per department

Task 3: Login System with Validation
Description: Develop a Login Page:
Validate inputs using JavaScript
Check credentials from database
Show error messages dynamically
Real-Time Usage: Authentication systems for any web application.

Task 4: Order Management using Joins
Concepts Used: Joins, Subqueries, ORDER BY, CSS layout
Create tables for:
Customers
Orders
Products
Display a customer order history using JOIN queries and a subquery to find:
Highest value order
Most active customer
Real-Time Usage: E-commerce and retail systems.

Task 5: Transaction-Based Payment Simulation
Description: Simulate an online payment process:
Deduct balance from user account
Add amount to merchant account
Use COMMIT on success, ROLLBACK on failure
Real-Time Usage: Banking and digital payment applications.

Task 6: Automated Logging using Triggers & Views
A trigger that logs every INSERT or UPDATE
A view that shows daily activity reports
Real-Time Usage: Audit logging in enterprise databases.

Task 7: Interactive Web Form with Events & Functions
Description: Build an interactive feedback form:
Validate inputs on keypress
Highlight fields on mouse hover
Show confirmation on double-click submit
Use JS functions for reusable validation logic
Real-Time Usage: Customer feedback and survey systems

Task 8: Create a simple Employee Management module using Spring Core. Demonstrate Inversion of Control and Dependency Injection using annotations such as @Component and @Autowired. Use BeanFactory to manage beans and store employee data in memory.

Task 9: Develop a basic Spring MVC application using annotation-based configuration.

Create a controller to accept user requests and display employee details using the MVC flow without XML configuration.

Task 10: Build a Student CRUD application using Spring Boot. Map the database table using JPA annotations such as @Entity, @Id, @Table, and @Column. Perform Create, Read, Update, and Delete operations using a relational database.

Task 11: Implement a Data Access Layer using Spring Data JPA. Use JpaRepository and custom query methods to retrieve student records based on conditions such as department or age, and include sorting and pagination.

Task 12: Design and develop a RESTful API for Product Management using Spring Boot. Implement GET, POST, PUT, and DELETE methods and return JSON responses. Test the APIs using a REST client.

Task 13: Add exception handling and validation to the RESTful application. Implement global exception handling and apply validation annotations to ensure correct input data and meaningful error responses.

Task 14: Design a simple microservices-based system by splitting a monolithic application into at least two independent services. Apply microservices design principles such as loose coupling, single responsibility, and independent deployment.

Task 15: Implement Service Registry and Discovery using a tool such as Eureka. Register multiple microservices and enable dynamic service lookup instead of hard-coded service URLs.

Task 16: Configure an API Gateway to route client requests to appropriate microservices. Implement basic load balancing so that requests are distributed across multiple instances of a service.

Task 17: Enable inter-service communication using REST-based communication. Create one microservice that consumes REST APIs exposed by another microservice and handles responses and failures gracefully.

Task 18: Write unit test cases for microservices using a testing framework. Test service logic, REST controllers, and data handling independently to ensure reliability.

Task 19: Create a Git repository for a sample project and demonstrate basic Git operations such as initializing a repository, staging files, committing changes, and maintaining version history. Push the repository to GitHub and manage it using remote repositories.

Task 20: Implement branching strategies in Git by creating feature branches. Perform merging and rebasing operations, and intentionally create merge conflicts to understand and resolve them effectively.

Task 21: Implement a CI/CD pipeline using Jenkins, GitHub Actions, or GitLab CI/CD. Automate code checkout, build, test, and deployment steps, and demonstrate continuous integration and continuous deployment for the application

Task 22: Explore major cloud service providers such as AWS, Google Cloud, and Microsoft Azure. Identify and compare their core services related to compute, storage, databases, and networking, along with the pay-as-you-use pricing model.

Task 23: Apply Vibe Coding and Prompt Engineering techniques using a Generative AI tool. Design effective prompts to generate code snippets, cloud configuration templates, or application logic, and evaluate the quality and efficiency of the generated outputs for real-world business applications.

Task 24: Use Vibe Coding to build a complete cloud-based feature using Generative AI. Write iterative prompts to generate a REST API, cloud deployment steps, and security configurations. Refine the prompts based on the output quality, document prompt versions, and evaluate how prompt engineering improves code accuracy, scalability, and real-world usability.

PART – B

Project -1 : Design and develop a React JS based web application for Ticket Booking of an Internal Department Event (such as technical fest, or seminar).

The application should allow users (students/faculty) to view event details and book tickets online.

The system must include the following requirements:

Event Details Module

Display event name
Department name
Event date and time
Venue
Ticket price
Available number of tickets
Ticket Booking Module

Allow user to enter:

Name
Email ID
Department
Number of tickets required
Prevent booking more tickets than available
Display booking confirmation message after successful booking

Validation Requirements

All input fields must be mandatory
Email ID should be in valid format
Number of tickets should be a positive number
Display appropriate error messages for invalid inputs
React Concepts to be Used
Functional components
JSX
UseState hook for state management
Event handling
Conditional rendering
Output Requirements
Display updated available ticket count after booking

Show booking summary including:

User name
Event name
Number of tickets booked
Total amount
Exception / Error Handling
Handle invalid inputs gracefully
Display user-friendly error messages
Additional Features (Optional)
Reset booking form
Simple CSS styling for better UI
Separate components for Event Details and Booking Form

The application should be developed using React JS and run in a browser.
Proper component structure and code readability must be maintained.

Full Stack Project-2: Smart Campus Event Management System

Description: A web application for a college/university to manage campus events, workshops, and seminars. Students can view upcoming events, register, and provide feedback, while admins can create, update, or delete events.

Key Features:

Student Side:

Browse upcoming events (REST APIs + MVC pages)
Register for events with form validation
View registered events

Admin Side:

Add / Edit / Delete events (CRUD using Spring Boot + JPA)
Search events using filters (date, department, type)

View event registration statistics (aggregate functions)

Technical Highlights:

- Spring Core: Dependency Injection (@Autowired), Bean management
- Spring MVC: Annotation-based controllers (@Controller, @RequestMapping)

- Spring Boot: Embedded Tomcat server, REST API development (@RestController)
- Spring Data JPA: Entity mapping (@Entity, @Id, @Table), CRUD operations
- Validation: @NotNull, @Size for form inputs
- Security: Basic authentication for admin login
- Exception Handling: Global exception handling using @ControllerAdvice
- HTML5/CSS3 + Thymeleaf for frontend

Project-3: Project Idea: Job Portal Management System

Description: A web application that connects job seekers and employers. Students can create profiles, upload resumes, browse job listings, and apply online. Employers can post jobs, view applicants, and shortlist candidates.

Key Features:

User Side (Job Seeker / Student):

Register/login and maintain profile

Upload resumes (file storage in database or local folder)

Search and filter jobs by category, location, or experience

Apply for jobs and track application status

Receive notifications for shortlisted jobs

Admin/Employer Side:

Post new jobs (title, description, skills required, salary)

Edit or delete jobs

View applicants for posted jobs

Shortlist or reject applications

Technical Highlights:

Backend (Spring Boot + Spring MVC):

REST APIs for CRUD operations

Controllers using @RestController and @Controller

Validation using @NotNull, @Size

Exception handling with @ControllerAdvice

Database (Spring Data JPA):

Entities: User, Job, Application

Relationships: One-to-Many (Employer → Jobs), Many-to-One (Application → Job/User)

Queries: Custom finder methods, sorting, and filtering

Security:

Spring Security for login and role-based access (Student, Employer, Admin)

Password hashing for security

Frontend:

HTML5/CSS3 and Thymeleaf

Responsive forms and dashboards

Dynamic table filtering (JavaScript/ES6)

Optional Add-ons (Industry-Relevant):

File uploads for resumes (using MultipartFile)

Email notifications for job status

Simple dashboard analytics (number of applications, shortlisted candidates)

Resources Needed:

- MySQL / H2 Database
- Spring Boot starter projects (can use Spring Initializr)
- Any code editor: VS Code, IntelliJ Community
- Localhost server for testing (no paid cloud required)

H. Learning Resources

i. Text Books:

1. Ashok Shah & K. K. Saxena. (2019). Web Technologies. University Press.
2. Spring in Action, Craig Walls. (2018). Fifth Edition, Manning Publications.
3. Douglas Crockford. (2008). JavaScript: The Good Parts. O'Reilly Media.
4. Jaya V. & Ramesh P. (2024). Vibe Coding and Generative AI for Developers. TechPress Publications.

ii. Reference Books:

1. Kathy Sierra & Bert Bates. (2017). Head First Java (2nd ed.). O'Reilly Media.
2. Abraham Silberschatz, Henry F. Korth, & S. Sudarshan. (2019). Database System Concepts, Seventh Edition, McGraw-Hill Education.

iii. Online Resources:

1. W3Schools India – Free tutorials for HTML, CSS, JavaScript, and more with simple examples and runnable code. Website: https://www.w3schools.in
2. LearnVern – Complete Web Design Course in Hindi covering HTML5, CSS3, and JavaScript for beginners. Website: https://www.learnvern.com/complete-web-design
3. Engineers Coding Hub – Full Stack Projects with GitHub Code including Spring Boot+ HTML/CSS/JS tutorials. Website: https://engineerscodinghub.com/full-stack-projects-with-github-code.
4. Awesome Spring Boot – Curated GitHub repository with blogs, tutorials, and learning resources for Spring Boot.Website: https://github.com/RameshMF/awesome-spring-boot

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC225	**Problem Solving and Testing using Java**	**1**	**0**	**4**	**3**

A. Preamble

This course focuses on applying functional programming, algorithmic problem-solving patterns, object-oriented design, and efficient data handling techniques. It equips students with practical skills for competitive programming and software development through structured coding practices and systematic testing methods.

B. Prerequisite Course - Nil

C. Course Objectives

Learners are exposed to:

- Practice functional programming features for writing clean and efficient code.
- Apply algorithmic patterns to solve time- and space-constrained problems.
- Design reusable object-oriented solutions for programming challenges.
- Select suitable collections for efficient data handling and performance.
- Validate program correctness using testing, debugging, and logging tools.

D. Course Outcomes

Upon the successful completion of the course, students will be able to:

COs	Course Outcomes	K Level
CO1	Use functional programming constructs to process data effectively.	**K3**
CO2	Apply algorithmic thinking to develop optimized solutions.	**K3**
CO3	Design object-oriented structures for problem solving.	**K3**
CO4	Utilize Java collections to manage data with performance awareness.	**K3**
CO5	Implement testing and debugging techniques to ensure program reliability	**K3**
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of COs with Program Outcomes and Programme Specific Outcomes

Cos	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	2	1	2	1	3	-	-	-	-	-	-	2	3
CO2	3	3	2	2	2	-	-	-	-	-	-	3	3
CO3	2	2	3	1	2	-	-	-	-	-	-	2	3
CO4	2	2	2	1	3	-	-	-	-	-	-	2	3
CO5	1	2	2	2	3	-	-	-	-	-	-	1	3

3- High; 2-Medium; 1-Low

F. Course Contents

UNIT I – Functional Programming **L - 3 Hours**

Introduction to functional programming concepts – Lambda expressions – Functional interfaces – Method references – Streams API – Date and Time API (java.time package) – Collectors API - Custom comparators and sorting logic

UNIT II – Algorithmic Thinking & Competitive Problem Patterns **L - 3 Hours**

Constraint-driven solution design – Competitive problem patterns - Writing Efficient Code: Strassen’s matrix multiplication - Kadane’s algorithm for finding maximum subarray sum- String Handling – Knuth Morris Pratt pattern matching – Boyer Moore Algorithm for substring search – Manacher Algorithm for finding longest palindromic substring.

UNIT III – Object-Oriented Problem Solving **L - 3 Hours**

Object modelling for algorithmic problems - Class to class interaction, Class to method interaction - Inheritance- Designing reusable classes - Polymorphic behavior in algorithm design - Design Pattern - Interface-based design for extensibility - Immutable objects for safe computation - Exception-driven flow control in competitive environments.

UNIT IV – Advanced Data Handling using Collections **L - 3 Hours**

Advanced problem solving using Vector, LinkedList, Arrays, Collections, ArrayList, HashSet, HashMap – Frequency counting using HashMap - Nested Collections.

UNIT V – Advanced Software Testing **L - 3 Hours**

Black box vs White Box – Stress testing - JUNIT – Test Life cycle methods - assertEquals - TimeOut error - Advanced features — Parameterized Testing - Logging: Log4j framework utilization. Debugging: How to Debug in IDE, Launch/Attach Breakpoints/Conditional Breakpoints - Step In/Out/Over Variables. Edge case identification and stress testing.

TOTAL: 15 Hours

G. Laboratory Experiments

PART –A

Task 1. Given a list of integers, find the second highest number in the list using stream API. If there is no second highest number (all elements same or only one element), print -1.
Input Format:
First line: An integer N - the number of integers in the list.
Second line: N space-separated integers - the elements of the list.
Output Format:
Print a single integer - the second highest number. Print -1 if it does not exist.
Example 1:
Input:
5
4 2 8 1 5
Output:

5

Example 2:

Input:

3

5 5 5

Output:

-1

Task 2. Write the following methods that return a lambda expression performing a specified action:

PerformOperation isOdd(): The lambda expression must return true if a number is odd or false if it is even.

PerformOperation isPrime(): The lambda expression must return true if a number is prime or false if it is composite.

PerformOperation isPalindrome(): The lambda expression must return true if a number is a palindrome or false if it is not.

Input Format

Input is handled for you by the locked stub code in your editor.

Output Format

The locked stub code in your editor will print lines of output.

Sample Input

The first line contains an integer, (the number of test cases).

The subsequent lines each describe a test case in the form of space-separated integers:

The first integer specifies the condition to check for (for Odd/Even, for Prime, or for Palindrome). The second integer denotes the number to be checked.

Sample Input:

5

1 4

2 5

3 898

1 3

2 12

Sample Output

EVEN

PRIME

PALINDROME

ODD

COMPOSITE

Task 3. Given a list of persons, each with a name (String) and age (integer).Sort the persons alphabetically by name using method references.Filter the persons older than a given age limit using a static method reference.Convert all names to uppercase using an instance method reference.

Input Format

The first line contains an integer n — the number of persons.

The next n lines each contain a string name and an integer age separated by a space.

The last line contains an integer ageLimit — the age threshold for filtering.
Output Format
First line: sorted names (alphabetically), space-separated.
Second line: names of persons older than ageLimit, space-separated, in the order they appear in the input.
Third line: all names in uppercase, space-separated, in the original order.
Name contains only uppercase and lowercase English letters
Sample Input
3
Alice 23
Bob 30
Charlie 25
24
Sample Output
Alice Bob Charlie
Bob Charlie
ALICE BOB CHARLIE

Task 4. A list of events is given, each with a unique name and a date in the format yyyy-MM-dd. The goal is to process this list using Java's Date and Time API (java.time) to perform the following operations: first, sort all events chronologically by date; second, determine the earliest and latest events in the list; and third, given a month number, identify all events that occur in that specific month. The solution should handle parsing dates, sorting them, and filtering based on the month efficiently using the Date and Time API.
Sample Input:
5
EventA 2025-12-17
EventB 2024-01-10
EventC 2025-06-05
EventD 2024-12-31
EventE 2025-12-01
12
Sample Output:
EventB EventD EventC EventE EventA
EventB
EventA
EventE EventA

Task 5. A list of students is given, where each student has a name and total marks. The task is to sort students based on marks in descending order. If two students have the same marks, they must be sorted alphabetically by name. Finally, collect the top K students using Java Streams and Collectors API.
Input Format
The first line contains an integer N, the number of students.

The next N lines contain student details in the format:
name marks
The last line contains an integer K, representing the number of top students to display.
Output Format
Print the names of the top K students in sorted order, separated by space.
Sample Input

```
6
Arun 85
Bala 92
Charan 85
Divya 95
Esha 92
Farhan 88
3
```

Sample Output

```
Divya Bala Esha
```

Task 6.A list of sales records is given, where each record contains a category name, item name, and sales amount. The task is to calculate the total sales for each category and also identify the item that contributed the maximum sales amount within that category.
Input Format
The first line contains an integer N, representing the number of sales records.
The next N lines contain sales details in the format:
category itemName amount
Output Format
For each category, print the category name, total sales amount, and the item name with the highest sales in that category.Categories should be printed in the order of first appearance.
Sample Input

```
7
Electronics TV 50000
Electronics Mobile 40000
Furniture Sofa 30000
Electronics Laptop 70000
Furniture Chair 15000
Furniture Table 20000
Electronics Headphones 10000
```

Sample Output

```
Electronics 170000 Laptop
Furniture 65000 Sofa
```

Task 7.Given an array arr[], Find the prefix sum of the array. A prefix sum array is another array prefixSum[] of the same size, such that prefixSum[i] is arr[0] + arr[1] + arr[2] . . . arr[i].
Sample Input:
Input:

```
5
10 20 10 5 15
```

Sample Output:
Output: 10 30 40 45 60
Explanation: For each index i, add all the elements from 0 to i

Task 8.You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.
Return the max sliding window.
Example 1:
Input: nums = [1,3,-1,-3,5,3,6,7], k = 3
Output: [3,3,5,5,6,7]
Explanation:

```
Window position          Max
---------------          -----
[1  3  -1] -3  5  3  6  7     3
 1 [3  -1  -3] 5  3  6  7     3
 1  3 [-1  -3  5] 3  6  7     5
 1  3  -1 [-3  5  3] 6  7     5
 1  3  -1  -3 [5  3  6] 7     6
 1  3  -1  -3  5 [3  6  7]    7
```

Example 2:
Input: nums = [1], k = 1
Output: [1]

Task 9. Given an integer array nums sorted in non-decreasing order, remove the duplicates in-place such that each unique element appears only once. The relative order of the elements should be kept the same.
Consider the number of unique elements in nums to be k. After removing duplicates, return the number of unique elements k.
The first k elements of nums should contain the unique numbers in sorted order. The remaining elements beyond index k - 1 can be ignored.
Example 1:
Input: nums = [1,1,2]
Output: 2, nums = [1,2]
Explanation: Your function should return k = 2, with the first two elements of nums being 1 and 2 respectively.
Example 2:
Input: nums = [0,0,1,1,1,2,2,3,3,4]
Output: 5, nums = [0,1,2,3,4]
Explanation: Your function should return k = 5, with the first five elements of nums being 0, 1, 2, 3, and 4 respectively.

Task 10. Given a word pattern and a string text consisting of lowercase characters, the task is to return the count of substrings in text which are anagrams of the pattern.

Example 1:
Input : text = "forxxorfxdofr", pattern = "for"
Output : 3
Explanation : Anagrams present are for, orf and ofr. Each appears in the text once and hence the count is 3.
Example 2:
Input : text = "aabaabaa", pattern = "aaba"
Output : 4
Explanation : Anagrams present are aaba and abaa. Each appears twice in the text and hence the count is 4.

Task 11. You are given N integers. Find the count of numbers divisible by K.
Input Format
The input begins with two positive integers N,K. The next N lines contain one positive integer each denoted by Ai.
Output Format
Output a single number denoting how many integers are divisible by K.
Sample 1:
Input
7 3
1
51
966369
7
9
999996
11
Output
4

Task 12. Given a lowercase English string s of length n, encrypt it as follows:
Replace each contiguous group of the same character with the character followed by the lowercase hexadecimal representation of its length.
Reverse the entire modified string.
Return the encrypted string.
Note: All Hexadecimal letters should be converted to Lowercase letters.
Examples:
Input: s = "aaaaaaaaaaa"
Output: ba
Explanation: Count consecutive characters: "aaaaaaaaaaa" → "a11".
Convert frequency to hexadecimal: "a11" → "ab".
Reverse the string: "ab" → "ba".
Input: s = "abc"
Output: 1c1b1a
Explanation: Convert to character-frequency format: "abc" → "a1b1c1".
Convert counts to hexadecimal (no change as 1 remains 1).

Reverse the string: "a1b1c1" → "1c1b1a".

Task 13. Design a simple EntityRecord system to store and display information about multiple entities.
Each EntityRecord has the following attributes:
Enrollment_ID (integer, unique identifier)
Student (string, name of the entity or person)
Course (string, type or classification of the entity)
Instructor (string, person responsible or associated with the entity)
startDate (string in YYYY-MM-DD format, representing creation or start date)
duration (integer, representing duration in appropriate units, e.g., weeks or months)
The task is to implement a program that reads multiple records, stores them, and displays the details of all records in the order they were added.
Sample Input:
3
101 Alice JavaBasics John 2025-12-20 6
102 Bob PythonAdvanced Mary 2025-12-22 8
103 Charlie DataScience Mike 2026-01-05 10
Sample Output:
Enrollment_ID: 101, Student: Alice, Course: JavaBasics, Instructor: John, Start Date: 2025-12-20, Duration: 6 weeks
Enrollment_ID: 102, Student: Bob, Course: PythonAdvanced, Instructor: Mary, Start Date: 2025-12-22, Duration: 8 weeks
Enrollment_ID: 103, Student: Charlie, Course: DataScience, Instructor: Mike, Start Date: 2026-01-05, Duration: 10 weeks

Task 14. Design a simple vehicle rental system using inheritance.There is a base class Vehicle with the following attributes:
vehicleId (integer, unique ID)
modelName (string)
baseRent (double)
There are two subclasses that inherit from Vehicle:
Car: has an additional attribute seats (integer). The total rent for a car is calculated as baseRent + (seats * 100).
Bike: has an additional attribute engineCapacity (integer, in cc). The total rent for a bike is calculated as baseRent + (engineCapacity * 2).
The task is to implement these classes and write a program that reads multiple vehicle records (Car or Bike), stores them, calculates the total rent for each vehicle, and then displays the vehicle details along with the total rent.
Sample Input:
3
C 101 HondaCivic 2000 5
B 102 YamahaR15 500 150
C 103 ToyotaCorolla 2500 4

Sample Output:

```
Vehicle ID: 101, Model: HondaCivic, Total Rent: 2500.0
Vehicle ID: 102, Model: YamahaR15, Total Rent: 800.0
Vehicle ID: 103, Model: ToyotaCorolla, Total Rent: 2900.0
```

Task 15. .Design a payment processing system for an e-commerce platform using polymorphism.

There is a base class Payment with a method processPayment(double amount) which returns a confirmation message.

There are multiple subclasses that inherit from Payment:

CreditCardPayment: Charges a 2% processing fee on the amount.

PayPalPayment: Charges a flat fee of $1.50 per transaction.

UPIPayment: No extra fee.

The task is to implement these classes and a program that reads multiple payment instructions, creates the appropriate payment objects, and prints the result of processPayment() for each payment. Use polymorphism so that the main program can call processPayment() on a Payment reference without knowing the specific payment type.

Sample Input:

```
4
C 100
P 50
U 200
C 250
```

Sample Output:

```
Processed CreditCard payment: Total Amount = 102.00
Processed PayPal payment: Total Amount = 51.50
Processed UPI payment: Total Amount = 200.00
Processed CreditCard payment: Total Amount = 255.00
```

Task 16.Design a notification system for a web application using interface-based design to ensure extensibility.There is an interface Notification with a method:

void sendNotification(String message);

There are multiple implementations of this interface:

EmailNotification – sends the message via email.

SMSNotification – sends the message via SMS.

PushNotification – sends the message as a push notification.

The main program reads multiple notification requests, creates the appropriate Notification objects, and calls sendNotification() on them.

Goal: Use an interface so that new types of notifications (like WhatsAppNotification, SlackNotification) can be added in the future without modifying existing code.

Sample Input:

```
4
E Welcome_to_our_platform
S Your_OTP_is_1234
P New_offer_available
E Password_changed_successfully
```

Sample Output:

```
Sent Email notification: Welcome_to_our_platform
Sent SMS notification: Your_OTP_is_1234
Sent Push notification: New_offer_available
Sent Email notification: Password_changed_successfully
```

Task 17. Design an online order processing system where certain operations may fail and should be handled using exception-driven flow control.

Each Order has the following attributes:

orderId (integer, unique ID)

productName (string)

quantity (integer)

availableStock (integer)

The main program reads multiple orders and attempts to process each order.

If the quantity requested is greater than availableStock, an exception should be thrown and caught to display:

Order <orderId> failed: Insufficient stock

If the order is valid, print:

Order <orderId> processed successfully

The program must continue processing the remaining orders even if some orders fail.

Sample Input:

```
5
101 Laptop 2 5
102 Mouse 10 8
103 Keyboard 3 3
104 Monitor 1 0
105 Speaker 5 10
```

Sample Output:

```
Order 101 processed successfully
Order 102 failed: Insufficient stock
Order 103 processed successfully
Order 104 failed: Insufficient stock
Order 105 processed successfully
```

Task 18. Design a simple banking system where each account has a unique account number, an account holder name, and a balance. Implement a class Account to store this information. Create another class Bank that manages multiple accounts and provides the following operations: depositing money into an account, withdrawing money from an account, and transferring money between accounts. The Bank class should interact with Account objects to perform these operations safely, ensuring that withdrawals or transfers do not exceed the available balance and that accounts exist before performing any operation.

Your task is to implement these classes and write a program that can process a sequence of banking operations and produce the appropriate output for each operation.

Sample Input:

```
2
101 Alice 1000
102 Bob 500
5
DEPOSIT 101 200
WITHDRAW 102 100
TRANSFER 101 102 300
WITHDRAW 102 1000
TRANSFER 101 103 50
```

Sample Output:

```
Deposited 200 to Alice
Withdrawn 100 from Bob
Transferred 300 from Alice to Bob
Insufficient balance
Account not found
```

Task 19. Given an integer array nums and an integer k, return the k most frequent elements. You may return the answer in any order.

Example 1:
Input: nums = [1,1,1,2,2,3], k = 2
Output: [1,2]
Example 2:
Input: nums = [1], k = 1
Output: [1]
Example 3:
Input: nums = [1,2,1,2,1,2,3,1,3,2], k = 2
Output: [1,2]

PART – B

Task 20. Design a data structure that internally uses HashMap that follows the constraints of a Least Recently Used (LRU) cache.

Implement the LRUCache class:
LRUCache(int capacity) Initialize the LRU cache with positive size capacity.
int get(int key) Return the value of the key if the key exists, otherwise return -1.
void put(int key, int value) Update the value of the key if the key exists. Otherwise, add the key-value pair to the cache. If the number of keys exceeds the capacity from this operation, evict the least recently used key.
The functions get and put must each run in O(1) average time complexity.
Example 1:
Input

```
["LRUCache", "put", "put", "get", "put", "get", "put", "get", "get", "get"]
[[2], [1, 1], [2, 2], [1], [3, 3], [2], [4, 4], [1], [3], [4]]
```

Output

```
[null, null, null, 1, null, -1, null, -1, 3, 4]
```

Explanation

```
LRUCache lRUCache = new LRUCache(2);
lRUCache.put(1, 1); // cache is {1=1}
lRUCache.put(2, 2); // cache is {1=1, 2=2}
lRUCache.get(1);    // return 1
lRUCache.put(3, 3); // LRU key was 2, evicts key 2, cache is {1=1, 3=3}
lRUCache.get(2);    // returns -1 (not found)
lRUCache.put(4, 4); // LRU key was 1, evicts key 1, cache is {4=4, 3=3}
lRUCache.get(1);    // return -1 (not found)
lRUCache.get(3);    // return 3
lRUCache.get(4);    // return 4
```

Task 21. HashSet
For Given Number A, find if it's a COLORFUL number or not.
COLORFUL number:
A number can be broken into different contiguous sub-subsequence parts.
Suppose, a number 3245 can be broken into parts like 3 2 4 5 32 24 45 324 245.
And this number is a COLORFUL number, since product of every digit of a contiguous subsequence is different
Return 1 if A is a COLORFUL number, else return 0
Problem Constraints
0 <= A <= 109
Input Format
The first argument is an integer A.
Output Format
Return 1 if A is a COLORFUL number, else return 0
Example Input
A = 23
Example Output
1
Explanation
A = 23
2 3 23
2 -> 2
3 -> 3
23 -> 6
this number is a COLORFUL number since the product of every digit of a sub-sequence is different.
Output: 1

Task 22. Google needs to schedule meetings in different rooms. Each meeting has a startTime and endTime. You are given a list of meetings. Determine the minimum number of meeting rooms required so that all meetings can take place without overlap. Use TreeMap to track ongoing meetings efficiently
Sample Input:

3
30 75
0 50
60 150
Sample Output:
2

Task 23. Given a list of items, each with a name and a rating, the task is to:Filter items with rating greater than or equal to a specified threshold (e.g., 8.0).Sort the filtered items by rating in descending order.

If two items have the same rating, sort them lexicographically by name.The items should be stored and manipulated using ArrayList to allow dynamic operations.

Sample Input:
5
ItemA 8.8
ItemB 7.5
ItemC 8.6
ItemD 7.9
ItemE 8.4
Sample Output:
ItemA 8.8
ItemC 8.6
ItemE 8.4

Task 24.A streaming platform has a RecommendationEngine class that generates movie recommendations based on user preferences.

Write JUnit tests to cover:
Users with no watch history
Users with only one watched movie
Users with identical ratings across multiple genres
Users with extremely large watch history (stress test)
Ensure recommendations always return at least 5 movies if possible.
Input Format:
User user = new User("user1", new ArrayList<>());
Output Format:
JUnit Assertion Example:
List<Movie> recommendations = engine.getRecommendations(user);
assertNotNull(recommendations);
assertTrue(recommendations.size() >= 5);

Task 25. A company develops a MathUtils class containing methods for: Finding the maximum subarray sum, Computing prefix sums of an array, Performing frequency counting of elements. Write JUnit 5 unit tests for these methods covering:

Empty input arrays
Arrays with negative/positive numbers
Very large arrays (stress testing)

Input Format:
Arrays passed programmatically in test cases
Output Format:
Test report indicating pass/fail for all scenarios

Task 26.A UserAnalytics class processes a list of users with attributes like age, location, and activity score.The class uses Streams API and Lambda expressions to filter active users and compute average scores.A bug is causing incorrect averages for certain edge cases (empty list or all inactive users).Use IDE debugging tools (breakpoints, conditional breakpoints, step in/out) to identify and fix the bug.
Input Format:
Programmatically created list of user objects
Output Format:
Correct average score and fixed filtering

Task 27. A streaming platform logs events (e.g., play, pause, skip) for millions of users.Implement a LoggingService using Log4j:INFO for normal events,WARN for repeated fast events,ERROR for system failures (e.g., null event).Write JUnit tests to validate that correct log levels are generated for different scenarios.
Input Format:
Simulated event objects in test cases
Output Format:
Log output verification in console or file
A banking application allows multiple users to perform transactions (deposit and withdraw) on the same account simultaneously.Design an AccountService class with the following methods:
deposit(int amount)
withdraw(int amount)
getBalance()
The system must ensure thread safety, so that the final balance is always correct even when multiple threads access the account concurrently.Implement the AccountService class using proper synchronization techniques.Write JUnit 5 test cases to verify:Concurrent deposits do not cause incorrect balance updates.Concurrent withdrawals do not allow balance to go negative.Mixed deposit and withdrawal operations maintain data consistency.Perform a stress test using multiple threads (e.g., 100+ threads).Use IDE debugging tools and breakpoints to observe thread execution behavior.
Input Format
Initial balance is set programmatically.
Transactions are simulated using multiple threads in test cases.
Output Format
Final account balance after all operations.
JUnit test report showing pass/fail for concurrency scenarios.

Task 28. A recommendation engine uses TreeMap to track top-N items for users based on frequency and ratings.Implement stress tests to handle:Very large input size (10^6 items),Frequent updates and queries,Log execution time using Log4j for performance verification.

Input Format:

Simulated item additions and queries

Output Format:

Performance logs

Verification that top-N logic is correct

TOTAL: 60 Hours

H. Learning Resources

i. Text Books:

1. Raoul-Gabriel Urma, Mario Fusco, & Alan Mycroft. (2019). *Modern Java in Action*, Second Edition, Manning Publications.
2. Joshua Bloch. (2018). *Effective Java*, Third Edition, Addison-Wesley Professional.
3. Kent Beck & Erich Gamma. (2019). *JUnit in Action*, Third Edition, Manning Publications.
4. M. A. Weiss, Data Structures and Algorithm Analysis in Java, 3rd ed., Pearson Education, New Delhi, India, ISBN 978-8131760635.
5. M. T. Goodrich, R. Tamassia, M. H. Goldwasser, and S. Banerjee, Data Structures and Algorithms in Java, 6th ed., Indian Adaptation, Wiley India Pvt. Ltd., 2022, ISBN 978-9354247934.

ii. Reference Books:

1. Kathy Sierra & Bert Bates. (2017). Head First Java (2nd ed.). O’Reilly Media.
2. Steven S. Skiena. (2020). *The Algorithm Design Manual*, Third Edition, Springer.

iii. **Online Resources:**

1. “Competitive Programming - A Complete Guide”, [Online] Last Updated: 22.08.2025. Available: https://www.geeksforgeeks.org/dsa/competitive-programming-a-complete-guide/
2. Junit5 - [Online] Last Updated: 22.08.2025. https://www.geeksforgeeks.org/software-testing/introduction-to-junit-5/

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC226	Java Programming	3	0	2	4

A. Preamble

This course focuses on basics of java programming, object oriented programming, java collection and effective data handling techniques. It equips students with practical skills for competitive programming and software development through structured coding practices and systematic testing methods.

B. Prerequisite Course: Nil

C. Course Objectives

Learners are able to:

- Apply basic java programming for writing clean code.
- Implement object oriented programming skills and handle exception to solve problems.
- Demonstrate solutions using suitable collections and multi-threaded programming.
- Utilize java data base connectivity to manage data.
- Validate program correctness using testing, debugging, and explore java 8 features.

D. Course Outcomes

Upon the successful completion of the course, students will be able to:

COs	Course Outcomes	K Level
CO1	Explain the fundamentals of Java and Object Oriented Programming (OOP) concepts to solve problems.	K3
CO2	Apply the concepts of java inheritance and exception handling mechanisms for real world problems.	K3
CO3	Solve the problems using Java Collection and Threads	K3
CO4	Utilize JDBC to handle data with database.	K3
CO5	Implement testing techniques to ensure program reliability and learn java 8 features.	K3
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of COs with Program Outcomes and Programme Specific Outcomes

Cos	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	1										
CO2	1	3	3	1	1							1	1
CO3	1	3	2	2	2							1	1
CO4	1	3	3		2								2
CO5	1	3	3		3								2

3- High; 2-Medium; 1-Low

F. Course Contents

UNIT I – Basics of OOP and JAVA L - 9 Hours

Introduction to object-oriented programming - Features of Java - JVM- Keywords- Variables – Data types – Operators - Control statements -Command line arguments - Compile Time Error - Classes and Methods - Objects – Number based problems - Java Packages – Build-in packages - user-defined package - Access specifiers: private, protected, default, public - Constructors - Method Overloading - Type Casting - this keyword- Static – Arrays, Multidimensional Arrays - Array based problems.

UNIT II – Inheritance, Exception Handling and string L - 9 Hours

Basics of Inheritance - Forms of Inheritance - Super keyword – Final - Method Overriding – Abstraction: Abstract Classes - Interfaces. Exception Handling: Java Exception Hierarchy - Exception Types - Throwing and Catching exceptions - Declaring New Exception Types – custom exceptions. String class – Immutable String - String handling methods- String based problems - Runtime Time Errors.

UNIT III – Collection and Problem Solving L - 9 Hours

Java.lang.Object – Array of Objects – java.lang.Arrays – Collection: ArrayList, LinkedList, Vector, HashMap, TreeMap - Debugging and Bug Fixing – Logic Mistake – Infinite Loops – Timeout error. Thread - Life Cycle - Multi-Threaded Programming — Thread Synchronization.

UNIT IV – Data Base Connectivity L – 9 Hours

Establish Connection to Data Base– Performing CRUD operations on Database – Insert, Delete – Update – Select - Join – Commit and Rollback – Debugging the JDBC Code.

UNIT V –Testing and Java 8 features L - 9 Hours

Software Testing Life Cycle (STLC) - JUNIT - Introduction to Junit Annotations –- Configuring unit tests in IDE - Writing and executing unit tests- Java 8 features – Lambda expressions – Functional interfaces – Method references – Streams API – Date and Time API (java.time package) – Collectors API.

OTAL: 45 Hours

G. Laboratory Experiments

Part –A

Task 1.

The following requires knowledge on conditional statements.

IS EVEN? https://tests.mettl.com/authenticateKey/2bd025dc

IS ODD? https://tests.mettl.com/authenticateKey/dbdac2a9

Return last digit of the given number

https://tests.mettl.com/authenticateKey/454f012b

Return second last digit of given number

https://tests.mettl.com/authenticateKey/9f87004e

Sum of last digit of two given numbers

https://tests.mettl.com/authenticateKey/783a1fcf

Is N an exact multiple of M?

https://tests.mettl.com/authenticateKey/36c4ef58

Of given 5 numbers, how many are even?

https://tests.mettl.com/authenticateKey/8edbe922

Of given 5 numbers, how many are odd?

https://tests.mettl.com/authenticateKey/67147bd5

Of given 5 numbers, how many are even or odd?

https://tests.mettl.com/authenticatekey/607636d7

The following will require looping concepts and simple maths.

Is Prime?

https://tests.mettl.com/authenticateKey/b1efaa3d

Factorial of a number

https://tests.mettl.com/authenticateKey/8c1f2ae

Nth Fibonacci

https://tests.mettl.com/authenticateKey/f390cadf

Nth Prime

https://tests.mettl.com/authenticateKey/34fdaa41

Task 2. Given an array arr[], Find the prefix sum of the array. A prefix sum array is another array prefixSum[] of the same size, such that prefixSum[i] is arr[0] + arr[1] + arr[2] . . . arr[i].

Sample Input:

Input:

5

10 20 10 5 15

Sample Output:

Output: 10 30 40 45 60

Explanation: For each index i, add all the elements from 0 to i

Task 3. You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.

Return the max sliding window.

Example 1:

Input: nums = [1,3,-1,-3,5,3,6,7], k = 3

Output: [3,3,5,5,6,7]

Explanation:

```
Window position            Max
---------------            -----
[1  3  -1] -3  5  3  6  7    3
 1 [3  -1  -3] 5  3  6  7    3
 1  3 [-1  -3  5] 3  6  7    5
 1  3  -1 [-3  5  3] 6  7    5
 1  3  -1  -3 [5  3  6] 7    6
 1  3  -1  -3  5 [3  6  7]   7
```

Example 2:
Input: nums = [1], k = 1
Output: [1]

Task 4.
You are given an integer array of size N.
If an element is even, add it to the sum.
If an element is odd, subtract it from the sum.
Find the final adjusted sum.
Input:
N = 5
Array = 10 15 20 25 30
Output:
20
Explanation:
Even sum = 10 + 20 + 30 = 60
Odd sum = 15 + 25 = 40
Adjusted sum = 60 - 40 = 20

Task 5 - FIRST REPEATING ELEMENT POSITION
Problem Statement:
Given an integer array, find the position (1-based index) of the first element that repeats.
If no element repeats, print -1.
Input:
Array = 4 5 1 2 5 3
Output:
2
Explanation:
Element 5 is the first repeating element.
Its first occurrence is at position 2.
Input:
Array = 1 2 3 4
Output:
-1

Task 6: ROTATE ARRAY AND FIND MAX DIFFERENCE
Problem Statement:
Rotate the given array to the right by K positions.
After rotation, find the maximum absolute difference between any two adjacent elements.
Input:
Array = 3 8 9 7 6
K = 1
Rotated Array:
6 3 8 9 7
Output:
5

Explanation:
Absolute differences:
|6 - 3| = 3
|3 - 8| = 5
|8 - 9| = 1
|9 - 7| = 2
Maximum difference = 5

Task 7. Given an integer array nums sorted in non-decreasing order, remove the duplicates in-place such that each unique element appears only once. The relative order of the elements should be kept the same.

Consider the number of unique elements in nums to be k. After removing duplicates, return the number of unique elements k.

The first k elements of nums should contain the unique numbers in sorted order. The remaining elements beyond index k - 1 can be ignored.

Example 1:
Input: nums = [1,1,2]
Output: 2, nums = [1,2]
Explanation: Your function should return k = 2, with the first two elements of nums being 1 and 2 respectively.
Example 2:
Input: nums = [0,0,1,1,1,2,2,3,3,4]
Output: 5, nums = [0,1,2,3,4]
Explanation: Your function should return k = 5, with the first five elements of nums being 0, 1, 2, 3, and 4 respectively.

Task 8. Given a word pattern and a string text consisting of lowercase characters, the task is to return the count of substrings in text which are anagrams of the pattern.

Example 1:
Input : text = "forxxorfxdofr", pattern = "for"
Output : 3
Explanation : Anagrams present are for, orf and ofr. Each appears in the text once and hence the count is 3.
Example 2:
Input : text = "aabaabaa", pattern = "aaba"
Output : 4
Explanation : Anagrams present are aaba and abaa. Each appears twice in the text and hence the count is 4.

Task 9: Give demonstration on the following exceptions**:**
A). Write a Java program to demonstrate ArrayIndexOutOfBoundsExceptionby accessing an array element outside its valid index range.
B. Write a Java program to demonstrate StringIndexOutOfBoundsExceptionby trying to

access a character at an invalid index in a string.

C. Write a Java program to demonstrate NullPointerExceptionby invoking a method on a null object.

D. 4 Write a Java program to demonstrate NumberFormatExceptionby converting a non-numeric string into an integer.

E. Write a Java program to demonstrate all the following exceptions using separate try–catch blocks:

* ArrayIndexOutOfBoundsException
* StringIndexOutOfBoundsException
* NullPointerException
* NumberFormatException

F. Write a Java program to create and demonstrate a user-defined exception.

Define a custom exception class and throw the exception when a user enters an invalid input (for example, age less than 18 or negative balance). Handle the exception using a try–catch block and display an appropriate error message.

Task 10. Given a lowercase English string s of length n, encrypt it as follows:

Replace each contiguous group of the same character with the character followed by the lowercase hexadecimal representation of its length.

Reverse the entire modified string.

Return the encrypted string.

Note: All Hexadecimal letters should be converted to Lowercase letters.

Examples:

Sample Input: s = "aaaaaaaaaaa"

Sample Output: ba

Explanation: Count consecutive characters: "aaaaaaaaaaa" → "a11".

Convert frequency to hexadecimal: "a11" → "ab".

Reverse the string: "ab" → "ba".

Input: s = "abc"

Output: 1c1b1a

Explanation: Convert to character-frequency format: "abc" → "a1b1c1".

Convert counts to hexadecimal (no change as 1 remains 1).

Reverse the string: "a1b1c1" → "1c1b1a".

Task 11. Design a simple EntityRecord system to store and display information about multiple entities. Each EntityRecord has the following attributes:

Enrollment_ID (integer, unique identifier)

Student (string, name of the entity or person)

Course (string, type or classification of the entity)

Instructor (string, person responsible or associated with the entity)

startDate (string in YYYY-MM-DD format, representing creation or start date)

duration (integer, representing duration in appropriate units, e.g., weeks or months)

The task is to implement a program that reads multiple records, stores them, and displays the details of all records in the order they were added.

Sample Input:
3
101 Alice JavaBasics John 2025-12-20 6
102 Bob PythonAdvanced Mary 2025-12-22 8
103 Charlie DataScience Mike 2026-01-05 10
Sample Output:
Enrollment_ID: 101, Student: Alice, Course: JavaBasics, Instructor: John, Start Date: 2025-12-20, Duration: 6 weeks
Enrollment_ID: 102, Student: Bob, Course: PythonAdvanced, Instructor: Mary, Start Date: 2025-12-22, Duration: 8 weeks
Enrollment_ID: 103, Student: Charlie, Course: DataScience, Instructor: Mike, Start Date: 2026-01-05, Duration: 10 weeks

Task 12. Given an integer array nums and an integer k, return the k most frequent elements. You may return the answer in any order.
Example 1:
Input: nums = [1,1,1,2,2,3], k = 2
Output: [1,2]
Example 2:
Input: nums = [1], k = 1
Output: [1]
Example 3:
Input: nums = [1,2,1,2,1,2,3,1,3,2], k = 2
Output: [1,2]

Task 13- Problem solving (This will require looping concepts and arrays).
Number of Primes in a specified range
https://tests.mettl.com/authenticateKey/87c41143
All Digits Count
https://tests.mettl.com/authenticateKey/ed6b4da
Unique Digits Count
https://tests.mettl.com/authenticateKey/b7aac4a5
Non-Repeated Digits' Count
https://tests.mettl.com/authenticateKey/e46500f5

Task 14 - Thread based problems
A). Creating Threads
Write a Java program to create two threads:
One thread should be created by extending the Thread class.
The other thread should be created by implementing the Runnable interface.
Each thread should print its name and run for 5 iterations.
B. Thread Lifecycle Demonstration
Create a Java program that demonstrates different thread states:
New

Runnable

Running

Waiting / Timed Waiting

Terminated

Print the thread state at different points during execution.

C. Thread Sleep and Join

Write a Java program with two threads:

The first thread prints numbers from 1 to 5 with a delay of 1 second.

The second thread should start only after the first thread finishes execution.

Use appropriate thread methods to control execution.

D. Synchronization Problem

Create a Java program that simulates a bank account:

Multiple threads try to withdraw money from the same account.

The account has a limited balance.

Ensure correct balance updates using thread synchronization.

E. Producer–Consumer Problem (Medium)

Write a Java program to implement the Producer–Consumer problem using threads:

A producer thread adds items to a shared buffer.

A consumer thread removes items from the buffer.

The buffer has a fixed size.

Handle thread communication properly to avoid data inconsistency.

Task 15. HashSet based problems.

For Given Number A, find if it's a COLORFUL number or not.

Definition for COLORFUL number:

A number can be broken into different contiguous sub-subsequence parts.

Suppose, a number 3245 can be broken into parts like 3 2 4 5 32 24 45 324 245.

And this number is a COLORFUL number, since product of every digit of a contiguous subsequence is different. Return 1 if A is a COLORFUL number, else return 0

Problem Constraints 0 <= A <= 109

Input Format

The first argument is an integer A.

Output Format

Return 1 if A is a COLORFUL number, else return 0

Sample Input

A = 23

Sample Output

1

Explanation

A = 23

2 3 23

2 -> 2

3 -> 3

23 -> 6

this number is a COLORFUL number since the product of every digit of a sub-sequence is different.

Output: 1

Task 16. Your manager needs to schedule meetings in different rooms. Each meeting has a startTime and endTime. You are given a list of meetings. Determine the minimum number of meeting rooms required so that all meetings can take place without overlap. Use TreeMap to track ongoing meetings efficiently

Sample Input:
3
30 75
0 50
60 150
Sample Output:
2

Task 17. Given a list of items, each with a name and a rating, the task is to:Filter items with rating greater than or equal to a specified threshold (e.g., 8.0).Sort the filtered items by rating in descending order.

If two items have the same rating, sort them lexicographically by name.The items should be stored and manipulated using ArrayList to allow dynamic operations.

Sample Input:
5
ItemA 8.8
ItemB 7.5
ItemC 8.6
ItemD 7.9
ItemE 8.4
Sample Output:
ItemA 8.8
ItemC 8.6
ItemE 8.4

Task 18. Given a sequence of numbers representing values in a dataset, the task is to find the Longest Increasing Subsequence (LIS). A subsequence is a sequence derived from the original sequence by deleting zero or more elements without changing the order of the remaining elements. The LIS is the longest subsequence such that each number is strictly greater than the previous number.

For large sequences, a naive recursive solution may be inefficient or cause stack overflow. The task is to compute the length of the LIS efficiently using memory-efficient iterative techniques.

Sample Input:
6
3 10 2 1 20 4
Sample Output:
3

Part – B

Task 19. Design a simple vehicle rental system using inheritance. There is a base class Vehicle with the following attributes:

vehicleId (integer, unique ID)
modelName (string)
baseRent (double)

There are two subclasses that inherit from Vehicle:

Car: has an additional attribute seats (integer). The total rent for a car is calculated as baseRent + (seats * 100).

Bike: has an additional attribute engineCapacity (integer, in cc). The total rent for a bike is calculated as baseRent + (engineCapacity * 2).

The task is to implement these classes and write a program that reads multiple vehicle records (Car or Bike), stores them, calculates the total rent for each vehicle, and then displays the vehicle details along with the total rent. Handle the necessary exceptions.

Sample Input: 3

```
C 101 HondaCivic 2000 5
B 102 YamahaR15 500 150
C 103 ToyotaCorolla 2500 4
```

Sample Output:

```
Vehicle ID: 101, Model: HondaCivic, Total Rent: 2500.0
Vehicle ID: 102, Model: YamahaR15, Total Rent: 800.0
Vehicle ID: 103, Model: ToyotaCorolla, Total Rent: 2900.0
```

Task 20. Design an income tax calculation system. The Income Tax Calculation System is a console-based Java application. The system takes user details such as name, age, and annual income, determines the applicable tax slab, calculates the tax amount, and displays the result. Consider the Indian taxation for tax calculation. Handle the necessary exceptions.

SAMPLE INPUT 1:

```
Enter Name: Anitha
Enter Age: 28
Enter Annual Income: 650000
```

SAMPLE OUTPUT 1:

```
----- Income Tax Calculation Result -----
Name: Anitha
Age: 28
Annual Income: 650000
Applicable Tax Slab:
5% on income between 2,50,001 and 5,00,000
20% on income between 5,00,001 and 6,50,000
Total Tax Amount: 42500
Net Income After Tax: 607500
```

SAMPLE INPUT 2 (NO TAX CASE):

```
Enter Name: Ramesh
Enter Age: 45
Enter Annual Income:
240000
```

SAMPLE OUTPUT 2:

```
----- Income Tax Calculation Result -----
Name: Ramesh
Age: 45
Annual Income: 240000
Applicable Tax Slab: No Tax
Total Tax Amount: 0
Net Income After Tax: 240000
```

SAMPLE INPUT 3 (EXCEPTION CASE – INVALID INPUT):

```
Enter Name: Suresh
Enter Age: 30
Enter Annual Income: abc
```

SAMPLE OUTPUT 3:

```
Error: Invalid income value entered.
Please enter a numeric value for annual income.
```

Task 21: Skill Management and Job Recommendation System

The Skill Management and Job Recommendation System helps students understand what skills are required for different job posts such as Software Developer, Web Developer, Data Analyst, etc. Handle the necessary exceptions.

The system:

Takes user-selected job role

Compares it with required skills

Displays missing skills

SAMPLE INPUT 1:

```
Select Job Role:
Software Developer
Web Developer
Data Analyst
Enter your choice:
1
Enter number of skills you already know:
3
Enter skill 1:
Java
Enter skill 2:
SQL
Enter skill 3:
OOPS
```

SAMPLE OUTPUT 1:

```
----- Skill Recommendation Report -----
Selected Job Role: Software Developer
Skills you already have:
Java
```

```
SQL
OOPS
Skills you need to study:
Data Structures
Git
```

SAMPLE INPUT 2:

```
Select Job Role:
Software Developer
Web Developer
Data Analyst
Enter your choice:2
Enter number of skills you already know:2
Enter skill 1:HTML
Enter skill 2:CSS
```

SAMPLE OUTPUT 2:

```
----- Skill Recommendation Report -----
Selected Job Role: Web Developer
Skills you already have:
HTML
CSS
Skills you need to study:
JavaScript
SQL
Java
```

SAMPLE INPUT 3 (EXCEPTION CASE – INVALID JOB CHOICE):

```
Select Job Role:
Software Developer
Web Developer
Data Analyst
Enter your choice:5
```

SAMPLE OUTPUT 3:

```
Error: Invalid job role selected.
Please choose a valid option from the menu.
```

SAMPLE INPUT 4 (EXCEPTION CASE – INVALID SKILL COUNT):

```
Enter number of skills you already know: -2
```

SAMPLE OUTPUT 4:

```
Error: Number of skills cannot be negative.
Please enter a valid positive number.
```

TOTAL: 30 Hours

H. Learning Resources

i. Text Books:

1. Balaguruswamy E, Programming in java, Sixth Edition, Tata McGraw Hill, 2019.
2. Mark Allen Weiss, Data Structures and Algorithm Analysis in Java, 3rd ed., Pearson Education, New Delhi, India, ISBN 978-8131760635.
3. Raoul-Gabriel Urma, Mario Fusco, & Alan Mycroft. (2019). Modern Java in Action, Second Edition, Manning Publications.
4. Joshua Bloch. (2018). *Effective Java*, Third Edition, Addison-Wesley Professional.
5. Kent Beck & Erich Gamma. (2019). *JUnit in Action*, Third Edition, Manning Publications.
6. M. T. Goodrich, R. Tamassia, M. H. Goldwasser, and S. Banerjee, Data Structures and Algorithms in Java, 6th ed., Indian Adaptation, Wiley India Pvt. Ltd., 2022, ISBN 978-9354247934.

ii. Reference Books:

1. Kathy Sierra & Bert Bates. (2017). Head First Java (2nd ed.). O’Reilly Media.
2. Steven S. Skiena. (2020). *The Algorithm Design Manual*, Third Edition, Springer.

iii. **Online Resources:**

1. “Competitive Programming - A Complete Guide”, [Online] Last Updated: 22.08.2025. Available: https://www.geeksforgeeks.org/dsa/competitive-programming-a-complete-guide/

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC301	Data Structures Laboratory	0	0	2	1

A. Preamble

This course is designed to develop skills to design and analyze linear and nonlinear data structures. This laboratory gives a way for students to get hands-on experience in identifying and applying the suitable data structure for the given real-world problem. It enables them to gain knowledge in practical applications of data structures.

B. Prerequisite Course

10210CS101- Problem Solving using C

C. Corequisite Course

10211CC101- Data structures

D. Course Objectives

Learners are exposed to

- Introduce various techniques for representation of the data in the real world.
- Design and implement various data structure algorithms through ADT including List, Stack, Queues.
- Get familiarize and implement non linear data structures including Trees and Graphs
- Implement various techniques of sorting and searching algorithms.

E. Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Design and implement basic array and various linear data structure operations.	**K3**
CO2	Develop an appropriate algorithm using tree data structure to the specific problem.	**K3**
CO3	Utilize appropriate hashing techniques and develop solutions for a problem.	**K3**
CO4	Apply Graph data structure and implement appropriate algorithm for given problem.	**K3**
CO5	Implement appropriate sorting and searching techniques for solving the problem.	**K3**
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

F. Correlation of Cos with Program Outcomes and Programme Specific Outcomes

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	3	2							3	3	3
CO2	3	3	3	3							3	3	1
CO3	3	3	3	3							3	3	1
CO4	3	3	3	3							3	3	1
CO5	3	3	3	3							3	3	3

High - 3, Medium - 2, Low – 1

G. Course Contents

PART 1

Task 1: Array ADT

Using Array ADT perform the following:

i) Basic Operations in ADT: Creating an array, displaying of array elements, Inserting an element

ii) Problems related to: Finding duplicates, missing element, minimum difference and rotations.

Task 2: Linked List

Design, Develop and Implement the following tasks using LinkedList ADT.

i) Singly Linked List: Operations in Ascending Order

ii) Doubly Linked List: Get reversed in from the given order

iii) Circular Linked List: Choose the place in the end of the circle so that you are the last one.

Task 3: Stack ADT

Develop and implement the following tasks using Stack ADT:

i) Stack using Array

ii) Find all elements greater than their Right element

iii) Infix To Postfix Conversion

iv) Balanced Parenthesis

v) Tower of Hanoi

Task 4: Queue ADT

Using Queue ADT perform the following:

i) Basic Operations of Queue ADT Using Array

ii) Queue Implementation using Stack

iii) Circular Queue of Characters

Task 5: Binary search Tree

Design, Develop and Implement the following tasks using Binary search Tree.

i) Basic Operations of BST

ii) Tree Traversal with Recursion

iii) Tree Traversal without Recursion

iv) Find the Depth and Height of the tree

v) Find the MIN and MAX element of the tree.

Task 6: Hashing

Using Hashing Technique perform the following:

i) Basic Hashing Technique
ii) Linear Probing
iii) Separate Chaining
iv) Character Frequency Counting
v) Number Frequency Counting

Task 7: Graph

Design, Develop and Implement the following tasks using Graph Concept.

i) Read the adjacent Vertices from given Matrices
ii) Find the Shortest Path using Dijkstra's Algorithm
iii) Find Minimum Spanning Tree using Prim's algorithm (based on Priority Queue)
iv) Find Minimum Spanning Tree using Kruskal's algorithm - using Disjoint subsets and Union find algorithm
v) Topological Sorting
vi) Graph Traversal using Depth First Search
vii) Graph Traversal using Breadth First Search

Task 8: Sorting

Develop and implement the following sorting algorithms:

i) Insertion Sort
ii) Merge sort
iii) Quick Sort
iv) Bubble sort
v) Radix sort.

Task 9: Searching

i) Implementation of Linear Search
ii) Implementation of Binary Search

PART 2:

Use Cases

Use Case - 1: Railroad Car

Imagine a railroad switching circuit such as the one below. Railroad cars are given unique numbers, from 1 onwards. Cars come in from the right in a random order, and thegoal is to assemble the cars in numeric order on the left.For example, to assemble the cars in the sequence shown (2, 1, 3) the car numbered 2would be switched on to the siding. Then car numbered 1 would be moved on to thesiding and then to the left. Next car numbered 2 would be moved up from the siding andassembled behind car 1. Finally, car numbered 3 would be moved from the right to theleft, via the siding. Notice that the cars in the siding work as a stack, since if a car ismoved on to a nonempty siding the existing cars cannot be accessed until the new car isremoved.

Develop and implement an algorithm that will rearrange the cars in sequential order regardless of the order in which they appear on the left. Use the Linear data structure ADT for implementing the use case.

Use Case - 2: Card Shuffling

A new casino has recently opened in ChefTown - the capital of Chef Land. Amongst a lot of games, there are also a few games that are played with a pile of cards. In total, there are N cards in the pile. Let us enumerate them by positive integers 1, 2, ..., N in order they appear in the pile from top to bottom.
Obviously, the cards must be shuffled before the game. Our protagonist Alex, being a mathematician wants to analyze this process.
According to Alex, the card shuffling process in the casino is quite similar to how we do in the real life. Namely, it consists of the following:

The croupier takes a consecutive ""sub-pile"" of cards, starting from the L1-th card from the top and ending at the R1-th card from the top (both inclusive) and moves this sub-pile from the middle of the pile to the very top, not changing the order of the cards in the taken sub-pile. Then she takes a sub-pile, starting at the L2-th card and ending this one at the R2-th one from the top of the current pile and moves it to the top, and so on. At last, she takes a sub-pile, starting at the LM-th card and ending this one at the RM-th one from the top of the current pile and moves it to the top.
The process, described above, repeats Q times.
Alex wants to collect some data in order to be able to predict the order in which the cards will be lying in the pile after the shuffling, but he is not really good at working with big piles and large repetitions. Design, Develop and implement an algorithm to do the following:
i) For a given set of segments ([L1, R1], [L2, R2], ..., [LM, RM]) and a given permutation, Find the minimum Q such that after Q repetitions of the first step in the algorithm above, the permutation will look like this. It is given that this number won't ever be bigger than 1012.
ii) For a given set of segments ([L1, R1], [L2, R2], ..., [LM, RM]) and a given Q, Determine the order of the cards after the shuffling algorithm above.

Use Case - 3: Multiuser Environment

Linux is designing a new OS to operate in a multiuser environment. All the users would be connected to a central machine. This central machine can allocate or de-allocate resources to the connected users in order to maximize overall performance.

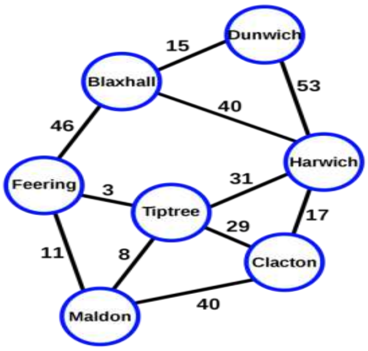
The OS has a limitation. If a particular program is running on more than 2 computers the central machine terminates this program on all the machines running it. To do this, the central machine assigns a unique identification number Z to each of the programs. The central machine runs a periodic check in which it obtains a list from each of users stating identification number of all the programs running on their systems. If the central machine finds that a particular program is running on more than 2 machines it terminates it on all machines.

This task of periodic checks and obtaining the lists has been implemented in the OS. In order to terminate the programs, the OS need to know in advance the exact number of programs it is going to terminate. So given these lists, design a program which can tell the OS the number of programs it is going to terminate.

Note: You need to specify only the number of ‘programs’ it is going to terminate and not the number of machines on which that program is running. For e.g. If a program is running on 4 computers, for the OS this counts as a single program which it needs to terminate.

Use Case - 4: Delivery Vehicles

A computer company in the Silicon Valley area (see Figure) needs to route delivery vehicles between cities on the shortest route. Recognize the appropriate algorithm and demonstrate the following tasks:



a. Modify the graph ADT to store weights in the arc nodes.
b. Implement a program, when given the start and destination and display the shortest route between them.

Total: 30 Hours

H .Learning Resources

i. Text Books:

1. Narasimha Karumanchi,” Data Structures and Algorithms made Easy”5th Edition, Career Monk Publication, Print Replica, 2020.
2. Seymour Lipschutz, “Data Structures with C”, Scham’s outlines, Tata McGraw Hill, 2017.

ii. Reference Books:

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, “Data Structures and Algorithms”, Pearson Education, First Edition, Reprint 2017.
2. Ellis Horowitz, Sartaj Sahni, Susan Anderson Freed, “Fundamentals of Data Structure in C”, Universities Press,2017.

iii. Online References:

1. “Data Structures using C” 2003- 2021. Accessed on: Apr. 15, 2021 [Online]. Available: http://www.academictutorials.com/data-structure/
2. “Linked List” 2015, Accessed on: Apr. 15,2021 [Online]. Available: http://www.c4learn.com/mcq/data-structure-mcq/linked-list-mcq-general-questions/
3. “Advanced Data Structures” 1999 – 2021. Accessed on: Apr. 15,2021 [Online]. Available: http://randu.org/tutorials/c/ads.php
4. “C Data Structure Fundamentals” 2021. Accessed on Apr.15,2021 [Online]. Available: http://www.zentut.com/c-tutorial/

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC312	Fundamentals of Computer Networks Laboratory	0	0	2	1

A.Preamble

This course provides students with hands on training regarding the design, troubleshooting, modeling and evaluation of computer networks. In this course, students are going to experiment in a passive cable connectivity, and learn about network design and troubleshooting topics such as: frame analysis, routing and routing table updating, route discovery, Switch Trunk Port Issues and many others.

B.Prerequisite Course

10210CS101 - Problem Solving using C

C.Corequisite Course

10211CC130– Fundamentals of Computer Networks

D.Course Objectives

Learners are exposed to

- Understand the working difference between straight cable and cross over cable.
- Gain core knowledge of Network layer routing protocols and IP addressing.
- Learn about basic security configuration and troubleshooting.

E.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Implement the network using network components	K3
CO2	Examine the frame analysis and routing in layer2 network.	K3
CO3	Apply the routing techniqueand create the subnetting in layer3 network.	K3
CO4	Demonstrate the Socket programming and datagram approaches.	K3
CO5	Explain the application layer protocols for a given real-time scenario.	K3
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

F.Correlation of Cos with Program Outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	2				3								
CO2	2	2	3		3								
CO3	2	2	3		3								
CO4	2	2	3		3								
CO5	2	2	3		3								

3- High; 2-Medium; 1-Low

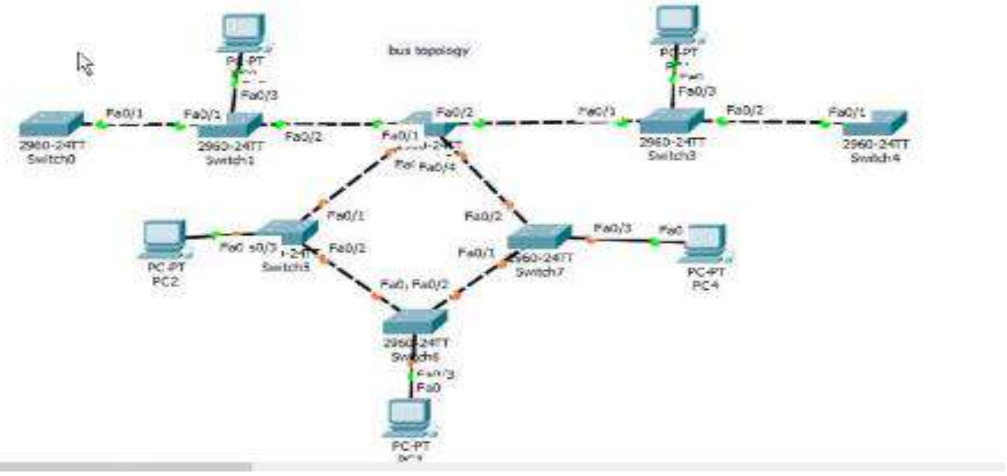
G.Course Contents

Part 1

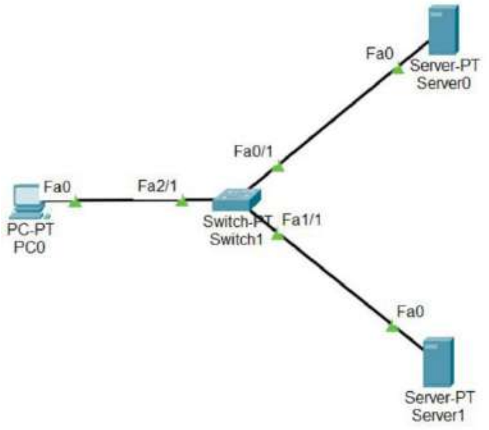
Physical Layer:

Task 1: Implement Peer-to-Peer network using network components: **CO1, K3**

Using raw Un Twisted Pair Cable (UTP) cables and connectors create Cross-over /straight-through cables as per the network requirements to fulfil the hybrid topology creation.



Task 2: Implement a client-server architecture using wireless medium.**CO1, K3**



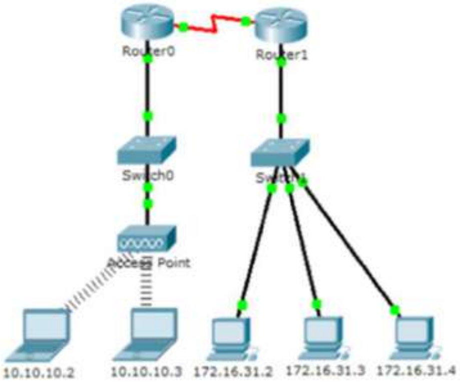
Data Link Layer:

Task 3: Frame Analysis**:** **CO2, K3**

A) Using PCAP tool capture the real time frame and analyses the all-header fields of the same.

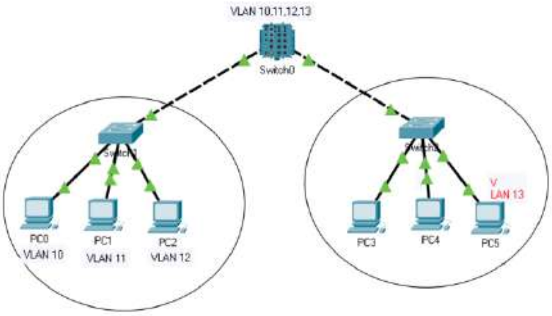
B) Configure a layer 2 switch to send a packet from one LAN (Local Area Network) to another LAN by Link Layer Address.

Task 4: Examine the MAC table using ARP **CO2, K3**



Task 5: Layer 2 Route Discovery: **CO2, K3**

Configure a layer 2 switch to send a packet from one LAN (Local Area Network) to another LAN by Link Layer Address.

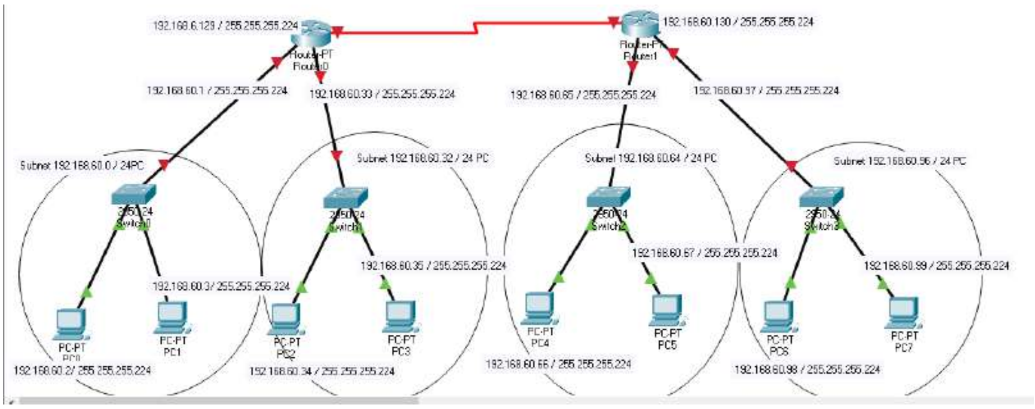


Execute the ping command from ‘PC2’ to ‘PC5’ successfully.

Network Layer:

Task 6: Subnetting **CO3, K3**

Calculate the range of network and host addresses and subnet mask for the given IP addresses (ex. 172.17.1.5/24) and construct a network for the same using packet tracer/NS2/NS3.



Task 7: Layer 3 Routing: **CO3, K3**

A) Consider networks A, B and C consist of 4 computers in each network are directly connected with Router ’R’.

Condition: I) network A and C can only communicate each other

II) Portion of Network B can communicate with Network A

Configure the above scenario using Cisco packet tracer or equivalent opensource tool

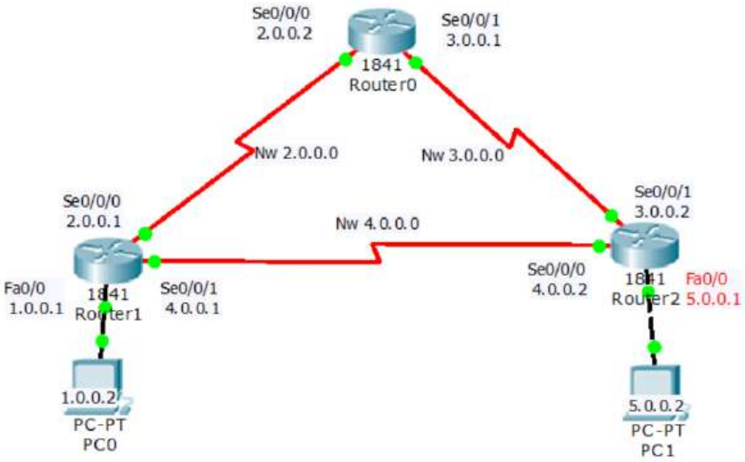
B) Using a wireless router configure the following

I) Device ID / name II) Device Password III) SSID IV) Wi-Fi Password

V) DHCP (Dynamic Host Configuration Protocol) Address range VI) MAC Binding.

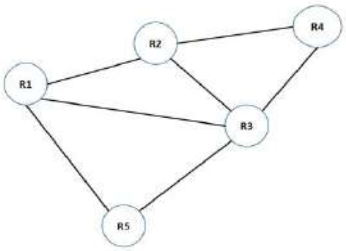
Task 8: Configuring RIP: **CO3, K3**

Implement to update routing information protocol table for the given network.



Task 9: Routing Table Update**:** **CO3, K3**

Implement to update routing table IPV4 and IPV6 addressing table before and after route 2(R2) has been removed from network communication for the given network.



Task 10: Protocol Performance**:** **CO3, K3**

Implement Distance Vector and Link State Routing Protocol using packet tracer with at least 4 routers. Justify the performance.

Transport Layer:

Task 11: Socket Programming**:** **CO4, K3**

Using TCP/IP sockets, create a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.

Task 12: Implement the datagram approaches. **CO4, K3**

Task 13: **Basic Security Configurations:** **CO4, K3**

Configure basic security consideration in a computer, Switch, Router and Firewall devices.

Task 14: Troubleshooting: **CO3, CO4, K3**

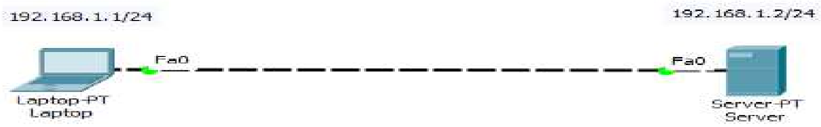
Study and Devise Solutions with appropriate commands to trouble shoot

i) Missing VLANs ii) Switch Trunk Port Issues iii) Switch Access Port Issues iv) Router Configuration Issues v) Remote management issue vi) restoring a managed switch with earlier configuration.

Application Layer:

Task 15: Remote Login: **CO5, K3**

Access the server's FTP service from the PC, capture the packet and analyze it.



Task 16: Implement the DNS services. **CO5, K3**

Task 17: Implement the web services using HTTP/HTTPS. **CO5, K3**

Task 18: Implement the mail service using SMTP, POP3, and IMAP. **CO5, K3**

Part 2

Use Case1: Designing LAN for an Operating Centre

Design an operational LAN network-1 with departments A, B are having 2 computers each, accessing the switch-1 located in Building 'B1' and the same part of departments which is operational at Building 'B2' are accessing the switch-2. Both switches are connected with Router 'R1' and the router ‘R1’ is serially connected with router 'R2'. R2 is directly connected with a PC ("stu VTU number")' using a Switch-3.

A Laptop 'Admin-PC' with Wireless connectivity to manage all network devices remotely and an application server 'Server-S1' is located in a network operating Centre connected with Router 'R1' via Switch-4 and is accessible to all departments.

Connectivity:

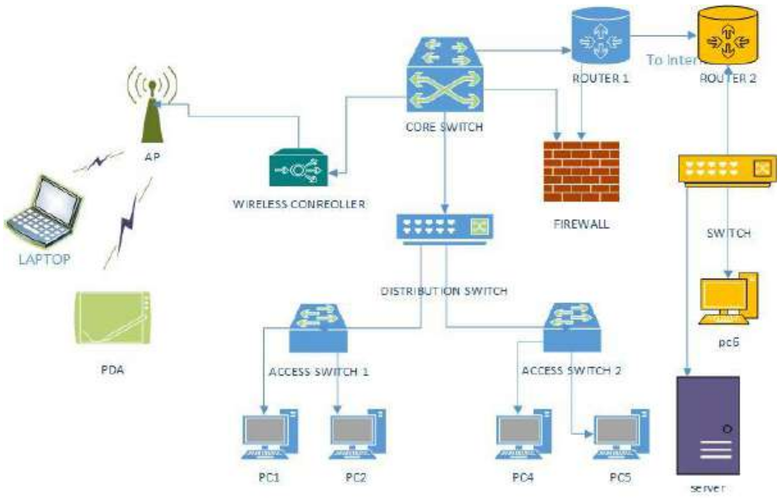
# All Network devices are connected to each other through FOC

# In-case of any additional devices required can be added

# Department C shall not communicate with Department A & B but can communicate with Server

# All not used Ports shall by Disabled and All switch requires login Password facility for remote users

# from laptop all switches need to be configured/Access for remote management



Use Case 2:

Create a Lan Network with 5 Wi-Fi device and 5 CCTV camera under a switch. Both traffic channels shall be separated and Only CCTV cameras need to be accessed from remote network beyond router. For that student may configure Port forwarding method.

Total: 30 Hours

F. Learning Resources

i. Text Books:

1. Behrouz Forouzan, “Data Communications and Networking”, Tata McGraw Hill, 5th Edition,2021.
2. Andrew S. Tanenbaum, David J Wetherall, Computer Networks, 5th Edition, Pearson Edu, 2016.

ii. Reference Books:

1. Kurose and K.W. Ross, “Computer Networking: A Top-Down Approach Featuring the Internet”, Addison-Wesley,2013
2. W. Stallings, “Data and Computer Communication”, PHI, 10th Edition,2017.
3. Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud,5th Edition,2015

iii. Online References:

1. “Networking Fundamentals” CISCO [online] 2019, https://www.cisco.com/c/dam/global/fi_fi/assets/docs/SMB_University_120307_Networking_Fundamentals.pdf

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC304	Operating Systems Laboratory	0	0	2	1

A.Preamble

The Operating Systems Laboratory is a course that will teach students about practicing Unix commands, Shell programming, System calls, working with process management, different CPU scheduling algorithms, Deadlock handling methods and apply Memory management techniques.

B.Prerequisite Course

10211CC101- Data Structures

C.Corequisite Course

10211CC103–Operating Systems

D.Course Objectives

Learners are exposed to

- Practice on Unix commands and Shell programming.
- Illustrate System calls implementation and process management techniques
- Demonstrate different CPU scheduling algorithms and Deadlock Handling methods
- Apply Various Memory and Disk scheduling techniques

E.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K-Level
CO1	Develop shell scripts and system calls.	K3
CO2	Apply the process management techniques and scheduling algorithms for the given problem.	K3
CO3	Make use of Banker's algorithm to detect and avoid dead lock.	K3
CO4	Implement various memory management strategies and page replacement algorithms.	K3
CO5	Implement the various disk scheduling algorithms.	K3
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

F.Correlation of COs with Program outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	1											
CO2	3	3	3	1								3	
CO3	3	3	3	3								3	
CO4	3	3	3									3	
CO5	3	3	3									3	

High-3, Medium-2, Low-1

G.Laboratory Experiments:

PART-1

Task 1. Practice on **UNIX Commands**

- For the given task find out the correct command and produce the output
- Ex: create 100 directories named from 1 to 100

Task 2. Interpret **Shell programming**

- For the given problem statement, write shell program.
- Find out the appropriate shell script to get the output.
- Implement the shell program to perform I/O operations on file.

Task 3. Illustrate **System calls** of UNIX operating system: fork, exec, getpid, exit, wait, close.

- Implement the appropriate program to implement the given system calls.
- Files created automatically by using system call implementation.
- Use relevant system calls in the implementation.

Task 4. Interpret **I/O System calls** of UNIX operating system (open, read, write, etc).

- Create the file by using appropriate system call in C program
- Perform read operation in the mentioned file and print it on screen
- Read input from user and write the same to the file.
- Copy content from one file, write into another file and display the content of destination file in display.

Task 5. Demonstrate **Inter-Process Communication** (**using shared memory**,pipes or message queues).

- Create number of processes and pass information between them.
- Pass information between parent and child process using pipes.
- Use shared memory concept, perform read and write operation by the process to communicate between them.

Task 6. Use of **semaphores** (using UNIX system calls)

Write program to

- Implement producer-consumer environment using common buffer.
- Implement readers-writers problem.
- Use appropriate semaphore variable for the given problem statement to avoid synchronization problems.

Task 7. Practice on CPU Scheduling algorithms: **FCFS, SJF, PRIORITY, ROUND ROBIN**

For the given scenario write C program to

- Find out the waiting time for each process.
- Find out the Turn around time of each process.
- Display the table which contains process names, burst time, arrival time, waiting time and turnaround time.
- Display average waiting time and Average turnaround time.

Task 8. Apply**Bankers Algorithm**to avoid Deadlock.

For the given scenario write C program to

- Calculate need matrix.
- Find out sequence of execution of process.
- Print the Sequence of execution and conclude the system in safe state or not.

Task 9. Interpret Memory allocation schemes like **First fit, Best fit and Worst fit.**

Write a C program to

- Simulate contiguous memory allocation techniques.
- Each allocation Techniques, find out the mapping of process with corresponding frames.
- Display mapping details.

Task 10. Illustrate **page Replacement algorithms: FIFO, LRU, OPTIMAL**

Write a C program to

- Implement different Page replacement algorithms, for each Algorithm,
- Show the contents of frames for each reference of a page.
- If page hit, display “No page Fault”
- Display the number of page faults at the end.

Task 11. Apply Disk scheduling algorithms :**FCFS, SSTF, SCAN,C-SCAN**

Write a C program to simulate Disk Scheduling techniques and for each algorithm

- Display the order of servicing the request.
- Calculate the number of read / write head movements.
- Display the number of movements.

<u>PART-2</u>

<u>Use Cases</u>

Use Case – 1:

Illustrate the given scenario by a program to implement

a) create a file “input.txt”
[one
Two
Three
Four
Five]

b) Copy “input.txt” to “output.txt”

c) Append to “output.txt”
[Six
Seven
Eight]

d) Delete “input.txt” file

Use Case – 2:

a) Demonstrate a program in C that creates a child process, waits for the termination of the child and lists its PID

b) Create a file named my file.txt that contains the following four lines :
Child 1 reads this line
Child 2 reads this line
Child 3 reads this line
Child 4 reads this line

Write a C program that forks four other processes. After forking the parent process goes into wait state and waits for the children to finish their execution. Each child process reads a line from the file my file.txt (Child 1 reads line 1, child 2 reads line 2, child 3 reads line 3 and child 4 reads line 4) and each prints the respective line. The lines can be printed in any order.

Use Case – 3:

Illustrate a program to coordinate the barber and the customers..A barbershop consists of a waiting room with n chairs, and the barber room containing the barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy, but chairs are available, then the Customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a program to coordinate the barber and the customers.

Use Case – 4:

A computer processor is given N tasks to perform ($1 \leq N \leq 50{,}000$). The i-th task requires T_i seconds of processing time ($1 \leq T_i \leq 1{,}000{,}000{,}000$). The processor runs the tasks as follows: each task is run in order, from 1 to N, for 1 second, and then the processor repeats this again starting from task 1. Once a task has been completed, it will not be run in later iterations. Determine, for each task, the total running time elapsed once the task has been completed.

Input

The first line of the input contains the integer N, and the next N lines contain the integers T_1 through T_N.

Output

Output N lines, the i-th of which contains an integer representing the time elapsed when task i has been processed.

Example

Input:

5

8

1

3

3

8

Output:

22

2

11

12

```
23
```

The second task is completed during the first iteration, finishing 2 seconds in. On the third iteration, the third and fourth tasks complete at 11 seconds and 12 seconds, respectively. Finally, on the eighth iteration, the first and last tasks complete at 22 seconds and 23 seconds, respectively.

Use Case – 5:

A single processor system has three resource types HDD, Printer and USB , which are shared by three processes. There are 5 units of each resource type. Consider the following scenario, where the column alloc denotes the number of units of each resource type allocated to each process, and the column request denotes the number of units of each resource type requested by a process in order to complete execution. Which of these processes will finish LAST? Develop a program to solve this.

1. P0
2. P1
3. P2

None of the above since the system is in a deadlock

	Alloc			**Request**		
	HDD	**Printer**	**USB**	**HDD**	**Printer**	**USB**
P0	1	2	1	1	0	3
P1	2	0	1	0	1	2
P2	2	2	1	1	2	0

Use Case – 6:

Interpret a C program to simulate page replacement algorithms
a) FIFO b) LRU c) LFU
A system uses 3 page frames for storing process pages in main memory. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below-Apply all replacement algorithms.
1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6
Also calculate the hit ratio and miss ratio.

Use Case – 7:

Demonstrate a C program to simulate the following contiguous memory allocation techniques
Consider six memory partitions of size 200 KB, 400 KB, 600 KB, 500 KB, 300 KB and 250 KB. These partitions need to be allocated to four processes of sizes 357 KB, 210 KB, 468 KB and 491 KB in that order.
Perform the allocation of processes using-

1. First Fit Algorithm

2. Best Fit Algorithm
3. Worst Fit Algorithm

Use Case – 8:

Implement a C program to simulate the following Disk Scheduling techniques

Input:

Suppose the order of request is- (82,170,43,140,24,16,190) print the order of visit and seek time

And current position of Read/Write head is : 50

a) FCFS b) SSTF

Total:30 Hours

H.Learning Resources

i. Text Book:

1. Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, "Operating System Concepts", 9th Edition, John Wiley and Sons Inc., 2016.

ii. Reference Books:

2. Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, "Operating System Concepts", 9th Edition, John Wiley and Sons Inc., 2016.
3. Richard Petersen, "Linux: The Complete Reference", 6th Edition McGrawHill Education 2017.

iii. Online References:

1. "Learn Unix" Accessed on: May 25, 2021 [Online]. Available:https://www.tutorialspoint.com/unix/index.htm
2. "Linux programming and scipting" June 18,2015. Accessed on: May 25, 2021 [Online]. Available:https://nptel.ac.in/courses/117/106/117106113/

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC305	**Microprocessors Laboratory**	**0**	**0**	**2**	**1**

A.Preamble

Thislaboratory course motivates the students to understand and develop assembly language programming skills by learning simple arithmetic, conditional programs. It teaches the students to realize the difference between microprocessor and microcontroller. Students are also exposed to interfacing techniques which has been the pathway for developing real time applications.

B.Prerequisite Course

10210EC201 – Basic Electronics & Digital Logic Design

C.Course Objectives

Learners are exposed to

- Understand 8086 Microprocessor architecture, instruction set.
- Develop assembly language programs using 8086 microprocessor.
- Implement processor interfacing with I/O devices.
- Develop applications using ARM processor.

D.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Develop programming skill using 8086 assembly language instructions.	**K3**
CO2	Identify the list of ARM processor related instructions and develop simple programs using it.	**K3**
CO3	Establish ARM processor interfacing with keyboard and display unit using embedded C programs.	**K3**
CO4	Realize UART data transmission between two ARM processor boards.	**K3**
CO5	Measure the temperature of an object using ARM processor.	**K3**
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of COs with Program outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3										1	
CO2	3	3			3							1	
CO3	3	3	3		3							1	
CO4	3	3	3		3							1	
CO5	3	3	3		3							1	

High-3, Medium-2, Low-1

F. Laboratory Experiments:

Part – 1

Task 1: 16bit arithmetic and logical operations using 8086 instruction set.
Construct assembly language programs for addition, subtraction, multiplication and division.

Task 2: String operations using 8086 instruction set
Develop programs for string concatenation and conversion of uppercase letters into lowercase letters and vice versa.

Task 3: Smallest and largest number using 8086 instruction set
Realize a program that is able to identify the smallest and largest number from the given set of data.

Task 4: Ascending and descending order using 8086 instruction set
Write assembly language program to arrange the given number from smallest to largest and vice versa.

Task 5: Data transfer using ARM processor
Employ an assembly language program which is able to transfer data from ARM processor register to memory or memory to ARM processor.

Task 6: Count of negative numbers in an array using ARM processor instructions
Formulate a logic to identify the list of negative numbers from the given numbers and develop the program using ARM processor instruction set.

Task 7: Factorial of positive integer N using ARM processor instructions
Calculate the factorial of a given numbers using ARM processor instructions.

Task 8: GCD algorithm using ARM processor instructions
Identify the greatest common divisor of given numbers by executing program developed using ARM processor instructions.

Task 9: Seven segment LED display
Control the display of an LED through an ARM processor.

Task 10: Key board and display interfacing using ARM processor kit development board
Realize the display of the information typed using a keyboard and processed by an ARM processor.

Task 11: UART device driver for data transmission
Experiment the process of data transfer that can be established between two ARM processor boards.

Task 12: Temperature measurement using ARM processor kit development board
Construct a data acquisition mechanism of any physical parameter like temperature using an ADC which is in-built in an ARM processor.

Part-2

Use Cases:

Use Case 1:

Develop an application which counts the number of customers entering into a supermarket for purchasing using 8086 microprocessor. IR sensor shall be used to identify the customer entering into the supermarket. 8086 Processor recognizes the output of IR sensor as a digital input and a suitable ALP is able to initiates the count as and when it receives a logic ‘1’.

Use Case 2 :

Implement a scheme to measure the distance between two cars that helps to avoid collision due to over speed. GP2D120 sensor can be used to measure the distance between two objects. Output of the sensor is analog signal. A/D converter in an ARM CORTEX processor recognizes this output as an input to it. Suitable Keil ‘C, coding implemented in the ARM processor converts the analog signal into digital and displays the result through LCD display.

Use Case 3 :

Interpret the given digital signal, develop a Keil ‘C’ program to implement in an ARM CORTEX processor for identifing the frequency of the signal and display its value in an LCD.

Use Case 4 :

Design a Keil ‘C’ program to generate a PWM signal based on the input value given by the user. The program must be implemented in an ARM CORTEX processor and the output shall be displayed by changing the ON / OFF condition of an LED. Timer in the ARM processor can change the width of the pulse depending the input applied to it. Frequency of the timer is the frequency of the ARM processor.

Total: 30 Hours

G. Learning Resources

i. Text Books:

1. Doughlas V. Hall, “Microprocessors and Interfacing”, TMH, Revised second edition, 2005.
2. Muhammad Tahir, Kashif Javed, “ARM Microprocessor Systems: Cortex-M Architecture, Programming, and Interfacing”, CRC Press, Taylor & Francis group, 2017.

ii. Reference Books:

1. Andrew N. Sloss, “ARM system developer’s guide – Designing and optimizing system software”, ELSEVIER, 2004.
2. Charles M. Gilmore, “Microprocessors: Principles and applications”, TMH, Second edition, 1995.
3. Jonathan W. Valvano, “Embedded microcomputer systems: Real time interfacing”, CENGAGE Learning, Third edition, 2012.
4. Steve Furber, “ARM system-on-chip architecture”, Pearson education, Second edition, 2015.

iii. Online References:

1. “Microprocessor interfacing”, Accessed on Sep 20, 2022 [Online], Available: https://onlinecourses.nptel.ac.in/noc20_ee11/preview.
2. “Microprocessors and Digital systems”, Accessed on Sep 21, 2022 [Online], Available: https://archive.org/details/microprocessorsd0000hall/page/n3/mode/2up.
3. “ARM processor architecture”, Accessed on Sep 21, 2022 [Online], Available: https://www.cs.ccu.edu.tw/~pahsiung/courses/ese/notes/ESD_03_ARM_Architecture.pdf

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC306	**Competitive Coding - I**	**0**	**0**	**2**	**1**

A.Preamble

This course provides the way to trace the code using pen and paper handy. Students are expected to understand the notions of syntax and semantics of programming languages. Also, the students are expected to carry out the tasks which in turn helps them to develop their logic and problem-solving skills using learning-by-doing approach.

B.Prerequisite Course

10210CS101– Problem Solving using C

C.Course Objectives

Learners are exposed to

- Understand in depth knowledge in tracing the code.
- Identify the syntax and semantic errors and to debug the program.
- Solve programming problems using basic number theory concepts
- Implement the concepts of linear and Nonlinear data structures for the real-world problem

D.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Trace and debug the program / code segment which includes syntax and semantic errors.	**K3**
CO2	Analyze the functional, logic and programming paradigm.	**K3**
CO3	Apply number theory and mathematical concepts to solve the computational problem.	**K3**
CO4	Implement algorithm for real word problems using elementary data structures.	**K3**
CO5	Analyze the given partial code / Pseudocode and find correct solution.	**K3**
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of Cos with Program Outcomes and Programme Specific Outcomes

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	3	3							3	3	
CO2	3	3	1	1							3	3	
CO3	3	3	3	3							3	3	3
CO4	3	3	3	3							3	3	3
CO5	3	3	3	1							3	3	

3-High, 2-Medium, 1-Low

F.Course Contents

The coursedescribes how programs work and how data is stored- Point out the Syntax and Semantic errors - Conceptual Questions in Programming- Solve programming problems implementing loops-Basic Number Theory 1, Basic Number Theory 2- Primality Tests- Data structure algorithms and finding complexity- Code Optimization -Partial Code Completion -Techniques for readability and ease of maintenance. -Concepts behind compiling, linking, OS, Hardware-Practice Problems for Placement Preparation

The above contents are covered under the following tasks:

Task1: Point out the Syntax and Semantic errors in the code snippet, and debug.
[Ref:T1–Pg.no:18,32,33,141-143,159-161,191-192,248,320-321,401,402, AppendixC]
Task2:ConceptualQuestionsinProgramming
[Ref: T1 – Pg.no:34,35,140,141,171,172,190,191,211,212,247248,269,271,319,320,400,401]
Task3:Solveprogrammingproblemsby implementingnecessaryloops.
[Ref:T1–Pg.no:48-53,66-72,84-86,102-106,120-124]
Task4:Basic NumberTheory 1
Tosolveprogrammingproblemswhichincludesthefollowingtopics:

1. **Modulararithmetic:**Propertieswithexample
2. **Modularexponentiation**
   a. Basicmethodanditstimecomplexitywithexample
   b. Optimizedmethodanditstimecomplexitywithexample

Task5:BasicNumberTheory1
Tosolveprogrammingproblemswhichincludesthefollowingtopics:

1. **Greatest CommonDivisor(GCD**)
   a. NaiveApproachanditstimecomplexitywithexample
   b. EuclidAlgorithmanditstimecomplexitywithexample
2. **ExtendedEuclideanalgorithm:**Basic methodand itstimecomplexity
3. **Modularmultiplicativeinverse**
   a. NaiveApproachanditstimecomplexitywithexample
   b. Euclideanalgorithmanditstimecomplexitywithexample
   c. Fermat’sLittleTheoremanditstimecomplexitywithexample

Task 6: Basic Number Theory 2
To solve programming problems which includes the concept of Prime numbers and its related properties.

a. **Naïve Approach** and its time complexity with example
b. **Sieve of Eratosthenes** and its time complexity with example

Task 7: Primality Tests
To solve programming problems which includes the concept of Prime numbers and its related properties.

1. **Sieve of Eratosthenes** with example
2. **Fermat’s Primality Testing** with example
3. **Miller-Rabin Primality** Testing with example

Task 8: Arrays -Introduction, memory allocation (with row/column major), operations (insert, delete, search) sorted and unsorted array, suffix array, subsequence and subarray.

Practice Problems:

i) Two Sum : -https://leetcode.com/problems/two-sum/
ii) Rotate Array :- https://leetcode.com/problems/rotate-array/
iii) Maximum Subarray :- https://leetcode.com/problems/maximum-subarray/
iv) Best Time to Buy and Sell Stock :-https://leetcode.com/problems/best-time-to-buy-and-sell-stock/
v) Container with most water:-https://leetcode.com/problems/container-with-most-water/

Additional Problems:

i) Contiguous array:-https://leetcode.com/problems/contiguous-array/
ii) Maximum product subarray:-https://leetcode.com/problems/maximum-product-subarray/
iii) Find the Duplicate Number:-https://leetcode.com/problems/find-the-duplicate-number/
iv) First Missing Positive:-https://leetcode.com/problems/first-missing-positive/

Task 9: String Processing, String functions (based on different language), String operations, Two algorithms for string pattern matching - Naïve approach and Robin karp approach

Practice Problems:

i) Reverse words in a string:-https://leetcode.com/problems/reverse-words-in-a-string/
ii) Valid Palindrome:-https://leetcode.com/problems/valid-palindrome-ii/
iii) Longest Palindromic Sunstring:-https://leetcode.com/problems/longest-palindromic-substring/
iv) Palindromic Substrings:-https://leetcode.com/problems/palindromic-substrings/
v) Subsequence Matching:-https://practice.geeksforgeeks.org/problems/subsequence-matching/0

Additional Problems:

i) Reverse String:-https://leetcode.com/problems/reverse-string/
ii) Reverse words in a string-III:-https://leetcode.com/problems/reverse-words-in-a-string-iii/
iii) Roman to Integer:-https://leetcode.com/problems/roman-to-integer/
iv) Good Substrings:-http://codeforces.com/problemset/problem/271/D
v) Pattern Finding:-https://www.spoj.com/problems/NAJPF/

Task 10: Bit Map-Introduction, XOR, AND, OR, right shift, left shift.

Practice Problems:

i) Reverse Bits: https://leetcode.com/problems/reverse-bits/
ii) Number of 1 bits: https://leetcode.com/problems/number-of-1-bits/
iii) Hamming Distance: https://leetcode.com/problems/hamming-distance/
iv) Single Number: https://leetcode.com/problems/single-number/
v) Single number II: https://leetcode.com/problems/single-number-ii/

Additional Problems:

i) Total Hamming Distance: https://leetcode.com/problems/total-hamming-distance/
ii) Counting Bits: https://leetcode.com/problems/counting-bits/
iii) Single Number III: https://leetcode.com/problems/single-number-iii/

Task 11: Recursion and its concepts

Practice Problems:

i) Fibonacci Number: https://leetcode.com/problems/fibonacci-number/
ii) Nth Tribonacci number: https://leetcode.com/problems/n-th-tribonacci-number/
iii) K equal sum subsets:https://leetcode.com/problems/partition-to-k-equal-sum-subsets/
iv) Permutations of string:https://practice.geeksforgeeks.org/problems/permutations-of-a-given-string/0
v) Counting Bits: https://leetcode.com/problems/counting-bits/

Additional Problems:

i) Total Hamming Distance: https://leetcode.com/problems/total-hamming-distance/
ii) Single number II: https://leetcode.com/problems/single-number-ii/
iii) Single Number III: https://leetcode.com/problems/single-number-iii/

Task 12: Elementary data structure algorithms-based Questions and finding complexity

S.No.	Data Structure Algorithms	Reference no- T2
12.1	Linked List	Problem no:2,3,11, 14,15,16,17,23,24,29
12.2	Stack	Problem no: 2, 3, 22, 23, 24, 25
12.3	Queue	Problem no: 1 to 6
12.4	Binary Tree	Problem no: 8, 10, 17, 18, 24, 27, 28
12.5	Binary Search Tree	Problem no: 49, 50, 51, 52, 56, 59
12.6	Sorting	Page no :246-255 Performance analysis and Comparison
12.7	Searching	Problem no: 1, 2, 36, 58, 59, 64, 67, 69

Task 13: Partial Code Completion [Ref: Online references]

Task 14: Practice Problems for Coding Preparation [Ref: T1 – Pgno:406-424]

Total: 30 Hours

G.Learning Resources

i.Text Books:

1. Yeshwant P. Kanetkar, "Let us C: Authentic Guide to C Programming Language",17th Edition, 2020, BPB Publications.
2. Narasimha Karumanchi, "Data Structures and Algorithms made Easy", 5th Edition, Career Monk Publication, Print Replica, 2020.

ii.Reference Books:

1. Marty Erickson, Anthony Vazzana, David Garth, "Introduction to Number Theory", Chapman and Hall, CRC; Copyright year 2016.
2. Thomas H.Cormen," Introduction to Algorithms",3rd Edition,2009, MIT Press Cambridge.

iii.Online References:

1. "Number Theory", Accessed on April 20, 2021 [Online]. Available: https://www.hackerearth.com/practice/math/number-theory/
2. "All Problem sets from Leet code", Accessed on Apr.20, 2021 [Online]. Available: https://leetcode.com/problemset/all/

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC307	**Competitive Coding -II**	**0**	**0**	**2**	**1**

A.Preamble

This course is recommended for students hoping to learn how to solve difficult problems that appear in multi-stage programming contests. The students will learn to design time and space efficient algorithms to solve challenging contest problems, and produce bug-free code under the pressure of time in contest. Students will learn to handle corner cases, invalid inputs, recursion conditions, memory leaks and test cases effectively. Learning by doing true end goal is to produce all-rounder computer scientists/ programmers who are much readier to produce better software and to face harder CS research problems in the future.

B.Prerequisite Course

10211CC306 – Competitive Coding - I

C.Course Objectives

Learners are exposed to

- Understand the basic concepts for solving real world problems
- Impart and implement the concepts of data structures
- Identify the basic properties of Trees and Graphs.
- Hands on Experience in analysis of algorithms for various algorithmic techniques
- Implement algorithm design methods such as dynamic programming and greedy method

D.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No’s	Course Outcomes	K - Level
CO1	Apply advanced Array, String and mathematical algorithms for problem solving.	**K3**
CO2	Demonstrate stack, queue and sorting algorithms to solve the problems.	**K3**
CO3	Implement classical algorithms for trees and graphs with backtracking concepts, being aware of their trade-offs in terms of complexity in time/space	**K3**
CO4	Analyse efficient algorithms to provide solution for computational problems using advanced data structures.	**K3**
CO5	Solve complex problems using dynamic programming and greedy algorithm.	**K3**
Knowledge Level (Based on revised Bloom’s Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of Cos with Program Outcomes and Programme Specific Outcomes

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	3	3							3	3	2
CO2	3	3	3	3							3	3	3
CO3	3	3	3	3							3	3	3
CO4	3	3	3	1							3	3	1
CO5	3	3	3	3							3	3	3

High - 3, Medium - 2, Low - 1

F.Course Contents

This course introduces advanced algorithms and data structures concepts useful for competing effectively in programming contests. This course provides familiarity with and proficiency in solving intermediate-difficulty algorithmic programming problems using dynamic programming, graph algorithms, mathematics, computational geometry, combinatorial games, and standard library data structures. It includes connectivity-based dynamic programming, multi-dimensional computational geometry, advanced string algorithms, advanced data structures, advanced graph and tree algorithms, linear programming, and satisfiability.

Task 1: Mathematics-Why math in coding? Greatest Common Divisor, Lowest Common Multiple, Euclidean Algorithm, Extended Euclidean Algorithm, Prime Numbers, divisibility of numbers and Sieve of Eratosthenes and Segmented Sieve, Modulo arithmetic, Modulo exponentiation and Modulo inverse, Math Probability, Counting, Game Theory, Group Theory, Generating functions, Permutation Cycles.

Practice Problems:

i) Sqrt(x):https://leetcode.com/problems/sqrtx/
ii) Trapping rain water: https://leetcode.com/problems/trapping-rain-water-ii/
iii) Permutation sequence: https://leetcode.com/problems/permutation-sequence/
iv) Ugly Number: https://leetcode.com/problems/ugly-number-ii/
v) Allocate Mailboxes: https://leetcode.com/problems/allocate-mailboxes/

Additional Problems:

i) Sum of square numbers: https://leetcode.com/problems/sum-of-square-numbers/
ii) Next permutation: https://leetcode.com/problems/next-permutation/
iii) Product of array:https://leetcode.com/problems/product-of-array-except-self/
iv) Trapping rain water: https://leetcode.com/problems/trapping-rain-water/
v) Find the median problem:https://www.hackerrank.com/challenges/find-the-median/problem
vi) Straight Line: https://leetcode.com/problems/check-if-it-is-a-straight-line/
vii) Number of digit one: https://leetcode.com/problems/number-of-digit-one/
viii) Largest component size by common factor:https://leetcode.com/problems/largest-component-size-by-common-factor/
ix) Reaching Points: https://leetcode.com/problems/reaching-points/
x) Beautiful Numbers: https://codeforces.com/problemset/problem/300/C

- xi) Number Expression:https://practice.geeksforgeeks.org/problems/check-if-a-number-can-be-expressed-as-xy1606/1
- xii) Pairs of Prime Numbers:https://practice.geeksforgeeks.org/problems/pairs-of-prime-number/0
- xiii) Fibonacci number:https://practice.geeksforgeeks.org/problems/check-if-the-number-is-fibonacci4654/1/
- xiv) Maximum product of three numbers:https://leetcode.com/problems/maximum-product-of-three-numbers/
- xv) Count all valid options:https://leetcode.com/problems/count-all-valid-pickup-and-delivery-options/
- xvi) Largest divisible subset:https://leetcode.com/problems/largest-divisible-subset/

Task 2: Advanced concept on Arrays – Two Pointer problem One maintains two pointers (indices) into either one array or two arrays, and move the pointers based on values at the pointers.

Practice Problems:

- i) Interval list intersections:https://leetcode.com/problems/interval-list-intersections/
- ii) Merge Intervals: https://leetcode.com/problems/merge-intervals/
- iii) Merge sorted array: https://leetcode.com/problems/merge-sorted-array/
- iv) Three sum: https://leetcode.com/problems/3sum/
- v) Smallest range covering element:https://leetcode.com/problems/smallest-range-covering-elements-from-k-lists/

Additional Problems:

- i) Longest substring without repeating characters:https://leetcode.com/problems/longest-substring-without-repeating-characters/
- ii) Push dominoes: https://leetcode.com/problems/push-dominoes/

Task 3: Advanced concept on String – String searching, Knuth–Morris–Pratt algorithm, Boyer–Moore string-search algorithm, Manacher's algorithm and z-function, suffix and prefix.

Practice Problems:

- i) Longest Prefix Suffix:https://practice.geeksforgeeks.org/problems/longest-prefix-suffix2527/1
- ii) Palindrome substring: https://codeforces.com/gym/101806/problem/Q
- iii) Password problem: https://codeforces.com/problemset/problem/126/B
- iv) Prefixes and suffixes: https://codeforces.com/problemset/problem/432/D
- v) Pattern find: https://www.spoj.com/problems/NAJPF/

Additional Problems:

- i) Palindrome:https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=26&page=show_problem&problem=2470
- ii) Chef and String: https://www.codechef.com/problems/CHSTR
- iii) Anthem of Berland: https://codeforces.com/contest/808/problem/G

Task 4: Stacks and Queues

Practice Problems:

- i) Sort characters by frequency:https://leetcode.com/problems/sort-characters-by-frequency/

ii) Valid Anagram: https://leetcode.com/problems/valid-anagram/
iii) Stack using Two queue :https://practice.geeksforgeeks.org/problems/stack-using-two-queues/1
iv) Bag of numbers : https://www.hackerearth.com/practice/data-structures/stacks/basics-of-stacks/practice-problems/algorithm/bag-of-numbers/
v) Disk tower : https://www.hackerearth.com/practice/data-structures/queues/basics-of-queues/practice-problems/algorithm/disk-tower-b7cc7a50/

Additional Problems:

i) Queries with fixed length problem: https://www.hackerrank.com/challenges/queries-with-fixed-length/problem
ii) Eerie Planet : https://www.hackerearth.com/practice/data-structures/queues/basics-of-queues/practice-problems/algorithm/weird-planet-2000a170/
iii) Number of atoms:https://leetcode.com/problems/number-of-atoms/

Task 5: Linked List-Single linked list, Doubly liked list, operations on linked list.

Practice Problems:

i) Add two Numbers: https://leetcode.com/problems/add-two-numbers/
ii) Remove nth Node:https://leetcode.com/problems/remove-nth-node-from-end-of-list/
iii) Merge two sorted list: https://leetcode.com/problems/merge-two-sorted-lists/
iv) Insertion sort list: https://leetcode.com/problems/insertion-sort-list/
v) Remove element:https://leetcode.com/problems/remove-linked-list-elements/

Additional Problems:

i) Sort List: https://leetcode.com/problems/sort-list/
ii) Reverse nodes in k groups: https://leetcode.com/problems/reverse-nodes-in-k-group/
iii) Remove duplicates:https://leetcode.com/problems/remove-duplicates-from-sorted-list-ii/
iv) Linked List cycle ii: https://leetcode.com/problems/linked-list-cycle-ii/
v) Odd Even linked list: https://leetcode.com/problems/odd-even-linked-list/
vi) Add two numbers II: https://leetcode.com/problems/add-two-numbers-ii/
vii) Swap nodes in pairs: https://leetcode.com/problems/swap-nodes-in-pairs/

Task 6: Sorting- Counting sort, Radix sort, Heap sort, Bucket sort

Practice Problems:

i) First non repeating character in a stream:https://practice.geeksforgeeks.org/problems/first-non-repeating-character-in-a-stream/0/
ii) LRU Cache: https://leetcode.com/problems/lru-cache/
iii) Finding pairs: https://www.hackerearth.com/practice/algorithms/sorting/counting-sort/practice-problems/algorithm/finding-pairs-4/
iv) Cube change :https://www.hackerearth.com/practice/algorithms/sorting/heap-sort/practice-problems/algorithm/cube-change-qualifier2/
v) Descending Weights: https://www.hackerearth.com/practice/algorithms/sorting/bucket-sort/practice-problems/algorithm/sort-the-array-5/

Additional Problems:

i) Binary Land: https://www.codechef.com/MAY20A/problems/BINLAND

ii) Divide Apples : https://www.hackerearth.com/practice/algorithms/sorting/heap-sort/practice-problems/algorithm/divide-apples/
iii) Shil and birthday present: https://www.hackerearth.com/practice/algorithms/sorting/counting-sort/practice-problems/algorithm/shil-and-birthday-present/

Task 7: Trees: Introduction, tree traversal (in-order, pre-order, post-order), Binary search tree

Practice Problems:

i) Longest Uni value Path:https://leetcode.com/problems/longest-univalue-path/
ii) Path Sum-III:https://leetcode.com/problems/path-sum-iii/
iii) Distinct count : https://www.hackerearth.com/practice/data-structures/trees/binary-search-tree/practice-problems/algorithm/distinct-count/
iv) MST revisited :https://www.hackerearth.com/practice/data-structures/trees/binary-search-tree/practice-problems/algorithm/mst-revisited-3f9d614c/
v) Vertical order:https://leetcode.com/problems/vertical-order-traversal-of-a-binary-tree/

Additional Problems:

i) Level order traversal:https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/
ii) Level order traversal -ii:https://leetcode.com/problems/binary-tree-level-order-traversal-ii/
iii) Preorder and inorder:https://leetcode.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/

Task 8: Graph - Introduction Adjacency Matrix and List, Depth First search, Breadth First Search, Manacher's algorithm

Practice Problems:

i) Decode string:https://leetcode.com/problems/decode-string/
ii) Cheapest Flights:https://leetcode.com/problems/cheapest-flights-within-k-stops/
iii) Network delay time:https://leetcode.com/problems/network-delay-time/
iv) Keys and rooms:https://leetcode.com/problems/keys-and-rooms/
v) Minimum cost to connect:https://leetcode.com/problems/minimum-cost-to-connect-two-groups-of-points/

Additional Problems:

i) Clone Graph:https://leetcode.com/problems/clone-graph
ii) Find the town judge:https://leetcode.com/problems/find-the-town-judge
iii) Bipartite:https://leetcode.com/problems/is-graph-bipartite/
iv) Path with minimum effort:https://leetcode.com/problems/path-with-minimum-effort/

Task 9: Backtracking-Introduction, Backtrack vs Divide and Conquer

Practice Problems:

i) Letter combination:https://leetcode.com/problems/letter-combinations-of-a-phone-number/
ii) Regular expression matching:https://leetcode.com/problems/regular-expression-matching/
iii) Generate parentheses:https://leetcode.com/problems/generate-parentheses/
iv) Combination sum:https://leetcode.com/problems/combination-sum/
v) Sequential digits:https://leetcode.com/problems/sequential-digits/

Additional Problems:

i) Permutations:https://leetcode.com/problems/permutations/
ii) Subsets:https://leetcode.com/problems/subsets/
iii) Path with maximum gold:https://leetcode.com/problems/path-with-maximum-gold/

Task 10: Heap Introduction, min-max heap, Segment tree &Trie

Practice Problems:

i) Find k pairs with smallest sums:https://leetcode.com/problems/find-k-pairs-with-smallest-sums/
ii) Last stone weight:https://leetcode.com/problems/last-stone-weight/
iii) Kth largest element in an array:https://leetcode.com/problems/kth-largest-element-in-an-array/
iv) K-closest points to origin:https://leetcode.com/problems/k-closest-points-to-origin/
v) Sliding window maximum:https://leetcode.com/problems/sliding-window-maximum/
vi) Range sum Query:https://leetcode.com/problems/range-sum-query-mutable/
vii) Maximum number of events:https://leetcode.com/problems/maximum-number-of-events-that- can-be-attended/
viii) Minimum number of increments:https://leetcode.com/problems/minimum-number-of- increments-on-subarrays-to-form-a-target-array
ix) Cost of Data : https://www.hackerearth.com/practice/data-structures/advanced-data-structures/trie-keyword-tree/practice-problems/algorithm/cost-of-data-11/
x) RandomGenerator:https://www.hackerearth.com/practice/data-structures/advanced-data-structures/trie-keyword-tree/practice-problems/algorithm/dexters-random-generator/

Additional Problems:

i) Skyline problem:https://leetcode.com/problems/the-skyline-problem/
ii) Count of smaller numbers:https://leetcode.com/problems/count-of-smaller-numbers-after-self/

Task 11: Dynamic programming - Introduction, Tabulation and memorization algorithms

Practice Problems:

i) Maximum sub array:https://leetcode.com/problems/maximum-subarray/
ii) Partition equal subset:https://leetcode.com/problems/partition-equal-subset-sum/
iii) Best time to buy and sell stock:https://leetcode.com/problems/best-time-to-buy-and-sell-stock/
iv) House robber:https://leetcode.com/problems/house-robber/
v) Minimum cost for tickets:https://leetcode.com/problems/minimum-cost-for-tickets/

Additional Problems:

i) Unique Paths:https://leetcode.com/problems/unique-paths/
ii) Minimum path sum:https://leetcode.com/problems/minimum-path-sum/

iii) Unique binary search:https://leetcode.com/problems/unique-binary-search-trees/
iv) Unique paths-ii:https://leetcode.com/problems/unique-paths-ii/
v) Longest increasing subsequence:https://leetcode.com/problems/longest-increasing-subsequence/
vi) Integer Break:https://leetcode.com/problems/integer-break/
vii) Longest palindromic substring:https://leetcode.com/problems/longest-palindromic-substring/

Task 12: Greedy algorithm -Introduction, algorithms based on greedy

Practice Problems:

i) Lemonade change:https://leetcode.com/problems/lemonade-change/
ii) Water bottles:https://leetcode.com/problems/water-bottles/
iii) Best time to buy and sell stock:https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-transaction-fee/
iv) Couples holding hands:https://leetcode.com/problems/couples-holding-hands/
v) Wildcard matching:https://leetcode.com/problems/wildcard-matching/

Additional Problems:

i) Partition labels:https://leetcode.com/problems/partition-labels/
ii) Queue reconstruction:https://leetcode.com/problems/queue-reconstruction-by-height/
iii) Reorganize string:https://leetcode.com/problems/reorganize-string/
iv) Jump game II:https://leetcode.com/problems/jump-game-ii/
v) Minimum number of taps:https://leetcode.com/problems/minimum-number-of-taps-to-open-to-water-a-garden/

Total: 30 Hours

G. Learning Resources

i. Text Books:

1. Antti Laaksonen, “Guide to Competitive Programming: Learning and Improving Algorithms Through Contests”, 2nd ed. 2020 Edition,UTiCS, Springer.
2. Antti Laaksonen, “Competitive Programmer's Handbook”, 3rd ed. 2020 Edition, Springer.
3. Steven Skiena, “The Algorithm Design Manual” 3rd Edition,2020 Springer.
4. Steven Halim and Felix Halim, “Competitive Programming, 3rd Edition”, Springer.

ii. Reference Books:

1. Johan Sannemo “Principles of Algorithmic Problem Solving”, 2nd eition, 2018.
2. Thomas H.Cormen, “Introduction to Algorithms”,3rd Edition,2009, MIT Press Cambridge.
3.Gayle Laakmann McDowell “Cracking the Coding Interview”, 3rd Ed, career cup publisher, 2015.

iii. Online References:

1.“Number Theory”, Accessed on April 20, 2021 [Online]. Available: https://www.hackerearth.com/practice/math/number-theory/
2.“All Problem sets from Leet code”, Accessed on April 20, 2021 [Online]. Available: https://leetcode.com/problemset/all/

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC310	IoT and Cloud Laboratory	0	0	2	1

A.Preamble

This course provides an overview of the Internet of Things (IoT) and Cloud Computing concepts, infrastructures and capabilities. This will help students gain the necessary knowledge to construct IoT systems and use cloud services for processing and storage of the data produced by the IoT devices. Emphasis will be placed on the architecture and design of IoT systems, the different technologies governing system implementation and the migration of the data to the Cloud for processing.

B.Prerequisite Course

10211CC104 - Modern Computer Architecture

C.Course Objectives

Students are able to

- Know the IoT architecture stack.
- Understand the different IoT platforms.
- Analyze the different IoT Protocols.
- Learn the cloud computing basics.
- Summarize role of Cloud Computing in IoT.

D.Course Outcomes:

Upon the successful completion of the course, students will be able to:

CO Nos.	Course Outcomes	K-Level
CO1	Build the basic design process using IoT simulation tools.	K3
CO2	Develop the IoT application using different hardware platforms.	K3
CO3	Illustrate the use ofvarious protocols in IoT applications.	K3
CO4	Demonstrate the use of cloud technology in IoT applications.	K3
CO5	Implement IoT-based systems for diverse applications	K3
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of Cos with Program Outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	2	2	2			1		1		2	1
CO2	3	2	2	2	2			1		1		3	1
CO3	3	2	2	2	2			2		2		3	2
CO4	3	2	2	2	2			2		2		3	1
CO5	3	3	3	2	3			2		3		3	1

3- High; 2-Medium; 1-Low

F.Laboratory Experiments

PART - 1

Task 1: Familiarization with Arduino/Raspberry Pi/Node MCU and perform necessary software installation.

Task 2: Familiarization with Fritzing and Thnikercad for design and fabrication of circuits.

Task 3: To interface LED/Buzzer with Arduino/ Raspberry Pi and write a program to implement traffic light system.

Task 4: To interface Digital sensor (IR/LDR) with Arduino/Raspberry Pi and write a program for object detection and find the distance between sensor and object.

Task 5: To interface DHT11 sensor with Arduino/Raspberry Pi and write a program to find the weather conditions and control the air conditioner.

Task 6: To interface motor using relay with Arduino/Raspberry Pi and write a program to control the direction and speed of the motor.

Task 7: To interface Bluetooth/WiFi with Arduino/Raspberry Pi and write a program to send sensor data to smartphone using Bluetooth.

Task 8: Write a program on Arduino/Raspberry Pi to push the data to cloud for detecting the fire event using temperature, light and smoke sensors.

Task 9: Write a program on Arduino/Raspberry Pi to upload/retrieve/visualize the sensor data to thingspeak/thingsio cloud for predicting the future data.

Task 10: Write a program on Arduino/Raspberry Pi to stream the sensor data with IoT core and use Bigquery to store data being streamed from IoT devices. Create Pub/Sub model for event driven systems.

Task 11: Use Cloud IoT Core for IoT based environmental monitoring application.

Task 12: Use Cloud IoT Core for IoT based Smart Light Controlling & Monitoring System.

PART - 2

Use Cases:

Use Case 1:

Implement a smart dust bin to detect level of Garbage, Temperature, Humidity, Gases like Ammonia, Carbon Monoxide and send a message with GPS coordinates to garbage collector using GSM module.

Use Case 2:

Implement a person centric health care management system to detect the medical events using IoT components and cloud

Use Case 3:

Implement a Smart home with IoT components and cloud for occupant monitoring.

Total: 30 Hours

G. Learning Resources

i. Text Books:

1. Alam, Mansaf, Kashish Ara Shakil, and Samiya Khan, eds. “Internet of Things (IoT): Concepts and Applications”. Springer Nature, 2020.
2. Mangla, Monika, “Integration of Cloud Computing with Internet of Things: Foundations, Analytics and Applications” John Wiley & Sons, 2021.

ii. Reference Books:

1. Lakhwani, Kamlesh, et al.,“Internet of Things (IoT): Principles, Paradigms and Applications of IoT”, Bpb Publications, 2020.
2. Parikshit N. Mahalle, Nancy Ambritta P., Gitanjali Rahul Shinde, Arvind Vinayak Deshpande, “The Convergence of Internet of Things and Cloud for Smart Computing”, CRC Press, 2021.

iii. Online References:

1. “IoT (Internet of Things) Wireless & Cloud Computing Emerging Technologies” Accessed on: April. 16, 2021 [Online]. Available: https://www.coursera.org/lecture/iot-wireless-cloud-computing/2-1-iot-architecture-part-1-K6pdR
2. “Practical IoT Concepts-Devices, IoT Protocols & Servers”, Accessed on: April. 16, 2021 [Online]. Available: https://www.udemy.com/course/practical-iot-devices-protocols-servers/
3. “Arduino Tutorials”, Accessed on: April. 16, 2021 [Online]. Available: https://www.arduino.cc/en/Tutorial/HomePage
4. “Thingspaek IoT Cloud”, Accessed on: April. 16, 2021 [Online]. Available: https://thingspeak.com/
5. “Tinkercad Tutorials”, Accessed on: April. 16, 2021 [Online]. Available: https://www.tinkercad.com/
6. “Industrial IoT on Cloud platform”, Accessed on April. 16, 2021 [Online]. Available:https://www.coursera.org/learn/iiot-google-cloud-platform

COURSE CODE	COURSE TITLE	L	T	P	C
10211CC313	**Problem Solving Techniques**	**0**	**0**	**2**	**1**

A.Preamble

In this course, problem solving techniques &logic building is practiced to the learners to make them think computationally by enhancing them to develop their logical thinking in which they can explore the usage and usefulness of problem solving to increase their efficiency to solve real world problems and to find a solution in the way that computer can also execute.

B.Prerequisite Course

10210CS101 – Problem Solving using C

C.Course Objectives

Learners are exposed to

- Expertise in breaking problems into parts and analyses the data to look for patterns by making sense of data.
- Identify different problem-solving styles.
- Find the methods appropriate for solving problems.
- Apply methods to specific problems.
- Identify, evaluate and synthesize information in a collaborative environment.

D.Course Outcomes:

Upon the successful completion of the course, students will be able to:

CO Nos.	Course Outcomes	K-Level
CO1	Select and use appropriate logic to solve problems effectively and creatively.	**K3**
CO2	Identify efficient approach for organizing and manipulating data using linear and non-linear concepts.	**K3**
CO3	Apply the decomposition techniques to breakdown the complex problems and find the logical similarities.	**K3**
CO4	Implement text processing techniques and abstraction methods to build optimized solution.	**K3**
CO5	Choose the problem-solving technique to develop pseudo code for real time problems.	**K3**
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of Cos with Program Outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	2							3	3	3
CO2	3	3	2	3							3	3	3
CO3	3	3	3	3							3	3	3
CO4	3	3	2	3							3	3	3
CO5	3	3	3	3							3	3	3

3- High; 2-Medium; 1-Low

F. Laboratory Experiments

List of Experiments:

LOGIC:

Task 1: Generation of Fibonacci Sequence. (TB2)
Task 2: Sine function Computation (TB2)
Task 3: Factorial Computation (TB2)
Task 4: Reversing the digits of an Integer (TB2).

DATA ORGANIZATION:

Task 5: Array Order Reversal (TB2)
Task 6: Find a maximum number in a Set (TB2)
Task 7: Remove duplicates from ordered array (TB2)
Task 8: Find k^{th} smallest element (TB2)

DECOMPOSITION:

Task 9:Breakdown of drawing smiley face (RB1-karl)
Task 10: Science project task breakdown (RB1-karl)
Task 11: Detection of Criminal Case (https://www.tes.com/teaching-resource/computational-thinking-activities-11636239)
Task 12:Two-way merge (TB2)

PATTERN RECOGNITION:

Task 13: DNA Sequencing (TB1)
Task 14: Linear Pattern Search(TB2)
Task 15: Sub Linear Pattern Search(TB2)

ABSTRACTION & TEXT PROCESSING

Task 16:Character to Number Conversion(TB2)
Task 17: Left and right Justification of Text(TB2)
Task 18: Text Line Editing(TB2)

FACTORING METHODS

Task 19:The Smallest divisor of an Integer(TB2)
Task 20:Prime Factors Computation.(TB2)
Task 21: Raising a number to a Large Power (TB2)

ALGORITHMIC THINKING AND PROBLEM SOLVING

Task 22: Sorting by Exchange, Selection& Insertion (TB2)
Task 23: Job Sequencing Problem with Deadlines. (https://www.techiedelight.com/job-sequencing-problem-deadlines/)
Task 24: Longest Monotone Subsequence (TB2)
Task 25: Recursive Quick Sort (TB2)

Total: 30 Hours

G. Learning Resources

i. Text Books:

1. David Riley and Kenny Hunt, “Computational thinking for modern solver”, Chapman &Hall/CRC, 2014 (TB1)
2. R.G. Dromey , “How to solve it by Computer”, PHI, 2008 (TB2).

ii. Reference Books:

1. Karl Beecher, “Computational Thinking - A beginner's guide to problem-solving and programming”, BCS Learning & Development Limited, 2017 (RB1)
2. Thomas H.Cormen, “Introduction to Algorithms”,3rd Edition MIT Press Cambridge.(RB2).

iii. Online References:

1. “Computational Thinking Activity”, Accessed on June 5, 2017 [Online]. Available:https://www.tes.com/teaching-resource/computational-thinking-activities-11636239.
2. “Computational Thinking Activity”, [Online]. Available: https://www.stem.family
3. “Teaching-Learning of Computational Thinking in K-12 Schools in India”, [Online]. Available: https://www.cse.iitb.ac.in/~sri/papers/CTE-Book-Chapter9-IYER_final.pdf.
4. “Computational Thinking in Education” Updated on December 11, 2019 [Online]. Available:https://www.it.iitb.ac.in/~sri/talks/CT-keynote-t4e2019.pptx.
5. “Algorithms & Approaches” [Online]. Available: https://www.techiedelight.com.

PROGRAM ELECTIVE

COURSE CODE	COURSE TITLE	L	T	P	C
10212CC123	Data Privacy and Security	2	0	0	2

A. Preamble

The purpose of this course is to cover the broad range of data security controls to protect data against compromises of confidentiality, integrity and availability by dealing with various security concepts like cryptography and network security, safeguarding of sensitive personal and corporate data against inadvertent disclosure. This course also includes the security and privacypolicies and legislations

B. Prerequisite Course

10211CC130 - Fundamentals of Computer Networks

C. Course Objectives

Students undergoing this course are expected to

- Perceive the Fundamentals of Data Security.
- Comprehend Data Privacy and Its Importance.
- Learn Data Anonymization Techniques.
- Explore Privacy-Preserving Test Data Generation.
- Understand Data Loss Prevention methods.

D. Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Understand the basics of Data Security	K2
CO2	Describe privacy protection mechanisms and its importance	K2
CO3	Implement the Anonymization technique to balance data and Privacy	K3
CO4	Explain the generation of Privacy Preserving Test Data	K3
CO5	Analyze Data Loss Prevention Policies and Strategies	K4
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of COs with Program outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1		2					2						
CO2	2	2										2	
CO3	2	2			3							2	
CO4	2	2			3		2					2	
CO5	3	1	3		2	1	1					2	1

3- High; 2-Medium; 1-Low

F. Course Contents

Unit 1 Data Security Fundamentals L-6 Hours

Introduction - The Need for Security - Threat - Attacks & Services - Security policies - cookie policy - Confidentiality - Integrity - Availability - Layering - Abstraction - Data Hiding - Security Controls - Access control structures - Security governance - Risk Analysis - Assessment - Response - Security planning.

Unit 2 Data Privacy and its Importance L-6 Hours

Data discovery and classification - need for sharing data - data subject access request - methods of protecting data - data breaches - policies, regulations, privacy models, importance ofbalancing data privacy and utility - disclosure methods: de-identification - suppression -statistical disclosure limitation(SDL) - coarsening - swapping - sampling - noise infusion - Emerging Applications: Social Network Privacy, Location Privacy, Query Log Privacy, Biomedical Privacy.

Case Study : Privacy issues in Healthcare and Social Network.

Unit 3 Data Anonymization L-6 Hours

Anonymization design principles, Multidimensional Data**:** Privacy Preservation methods - Group based anonymization: k-anonymity, i-diversity, t-closeness - Algorithm comparison - Complex Data**:** Privacy Preservation - Graph Data - Time Series Data - Longitudinal data - Synthetic DataGeneration - Dynamic Data Protection - Tokenization - Threats to anonymity.

Case Study : Theft of Unencrypted Laptop.

Unit 4 Privacy Preserving and Test Data Generation L-6 Hours

Privacy Preservation of Association Rule Mining - Privacy Preserving Clustering algorithms – Privacy - Preserving Synthetic Data Generation - Design for Privacy Preservation of Test Data Generation - Test Data Fundamentals - Utility of Test Data: Test Coverage - Privacy Preservationof Test Data - Quality of Test Data - Insufficiencies of Anonymized Test Data - Data Removal.

Unit 5 Data Loss Prevention L-6 Hours

Introduction and Objectives - Types of Data Loss - Key Components - Strategies for Data Inventories and Risk Assessments - DLP Policies, Procedures, and Controls - Deploying DLP Tools and Technologies - User Education and Awareness - Emerging Trends: Data ExfiltrationTechniques and Insider Threats.

Total: 30 Hours

H. Learning Resources

i.Text Books:

1. NatarajVenkataramanan, AshwinShriram, “Data Privacy: Principles and Practice”, Chapman and Hall/CRC, first edition, 2016. (ISBN No.: 978-1-49-872104-2). – (UNITI, II, III,IV).
2. Kris Hermans, “Mastering DLP: A Comprehensive Guide to Data Loss Prevention” Format: Kindle Edition. (ASIN No. : B0CD8D95S7). – (UNIT V).

ii. Reference Books:

1. Stephen Massey, “The Ultimate GDPR Practitioner Guide (2nd Edition): Demystifying Privacy & Data Protection”, Fox Red Risk; 2nd edition, 2020.
2. Micki Krause, Harold F. Tipton, “Data Protection and Privacy:”, Hart Publishing,(ISBN No. 978- 1509919345), 2017.
3. David Kim and Michael G.Solomon, “Fundamentals of Information Systems Security”, Jones and Bartlett Publishers, Inc, 2018.
4. VishwajitBarbudhe, Shraddha N Zanjat, Bhavana S Karmore, “Cryptography andNetwork Security Principles“, Lambert Academic Publishing, 2020.

iii. Online References:

1. William satallings,2021, williamstallings.com/Extras/Security-Notes/
2. Introduction to Cryptography and Computer Security,2015, http://cs.brown.edu/courses/csci1510/
3. Woodrow Hartzog, Northeastern University, Data Privacy Fundamentals, 2020, https://www.coursera.org/learn/northeastern-data-privacy.
4. "Data Loss Prevention" by The Art of Service, https://www.udemy.com/course/data-loss-prevention/?couponCode=ST3MT72524

COURSE CODE	COURSE TITLE	L	T	P	C
10212CC211	Artificial Intelligence Techniques	3	0	2	4

A. Preamble

The primary objective of this course is to introduce the basic principles, techniques, and applications of Artificial Intelligence. In this course students can explore the various Search techniques like basic, advanced and heuristic approach are universal problem-solving methods to solve a specific problem and provide the best result. Evolutionary algorithms are used to optimize the AI applications. Goal and constraint-based approach, Knowledge reasoning and inference helps the students to develop the decisions making strategies in intelligent agents and expert systems.

B. Prerequisite Course

10211CC101 - Data Structures

C. Course Objectives

Learners are exposed to

- Understand and apply the basic principles of artificial intelligence concepts towards problem solving, knowledge representation, reasoning, inference and learning.
- Explore the artificial intelligence techniques in intelligent agents, expert systems, artificial neural networks, Genetic algorithms, Heuristics approaches and Evolutionary algorithms models.

D. Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No’s	Course Outcomes	K – Level
CO1	Indentify the problem-solving techniques by using State Space search and heuristic search approach	K3
CO2	Use evolutionary and advanced search techniques to provide optimized solution.	K3
CO3	Solve unified planning approach and Constraint-satisfaction problems.	K3
CO4	Apply Logical knowledge representation for rules and fact-based approach.	K3
CO5	Build the logic of Uncertainty and Reasoning using resolution methods.	K3
Knowledge Level (Based on revised Bloom’s Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of COs with Program outcomes and Programme Specific Outcomes:

Cos	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3		1	1			1	1		1	3	
CO2	3	3	3	1	1			1	1		1	3	1
CO3	3	3	3	1	1			1	1		1	3	1
CO4	3	3	2	1	2			1	1		1	3	1
CO5	3	3	3	1	2			1	1		1	3	1

3-High; 2-Medium; 1-Low

F. Course Contents

Unit 1 Basic and Heuristic Search Techniques L-9 Hours

Artificial Intelligence – Introduction - Evolution of AI- State Space Search: Generate and Test- Simple Search- Depth First Search- Breadth First Search- Comparison of BFS and DFS- - Depth Bounded DFS- Depth First Iterative Deepening. Heuristic Search: Heuristic Functions- Best First Search- Hill Climbing- N-Puzzle Problem- Block-world Problem Local Maxima- Variable Neighborhood Descent- Beam Search- Tabu Search- Case Studies – Stock Market prediction, Weather Forecasting.

Unit 2 Evolutionary and Advanced Search Techniques L-9 Hours

Randomized Search and Emergent Systems: stochastic and evolutionary search algorithms-Simulated Annealing- - Emergent Systems- Ant Colony Optimization- Finding Optimal Paths: The Travelling Salesman Problem- Dynamic Programming- Algorithm A*- Iterative Deepening A* - Min-Max- Alpha-Beta Tree search – Monte Carlo Tree Search - Case Studies – Vehicle Routing and Map Navigation.

Unit 3 Planning and Strategies L-9 Hours

Planning: planning as search, partial order planning - A Unified Planning Framework-Forward and Backwards State Space Planning- Goal Stack Planning- Plan Space Planning-The STRIPS Domain- Constraint Satisfaction Problem: N-Queens-Cryptarithmetic- Graph Coloring- Constraint Propagation- Scene Labeling- Higher Order and Directional Consistency- Algorithm Backtracking: Graph Coloring - Case Studies – Robot traversal-Object recognition- Game Playing.

Unit 4 Knowledge Representation L-9Hours

Ontologies, foundations of knowledge representation- Logical representation -The Scheme-Frames- Semantic Net- Production rules -Scripts - Inheritance in Taxonomies- Description Logics- Formal Concept Analysis- Conceptual Graphs Knowledge reasoning - Reasoning about objects- relations – events- actions- time and space-reasoning with defaults- Reasoning about knowledge Case studies – chatbot system- recommendation system.

Unit 5 Knowledge Facets and Logic Inferences L-9 Hours

Knowledge Facets: Intelligent agents: reactive, deliberative, goal-driven, utility-driven, and learning agents- The Wumpus World. Logic and Inferences: Formal Logic- Resolution Method in Propositional Logic- Resolution method in First Order Logic- Deductive Retrieval – Horn

Clauses – Forward Chaining- Backward Chaining- Uncertain Knowledge- Probabilistic Reasoning- connection to logic- independence- Bayes rule- Bayesian networks- probabilistic inference, Case studies – Multi- Agent Decision Making System- Human Computer Interaction.

Total: 45 Hours (L)

G. Laboratory Experiments

PART – 1

Task 1: Implementation of Graph search algorithms (Breadth first search and Depth First Search) using following constraints.

BFS: Pick any node, visit the adjacent unvisited vertex, mark it as visited, display it, and insert it in a queue. If there are no remaining adjacent vertices left, remove the first vertex from the queue. Apply recursion concept to follow the above steps until the queue is empty or the desired node is found.

DFS: Pick any node. If it is unvisited, mark it as visited and recur on all its adjacent nodes. Repeat until all the nodes are visited, or the node to be searched is found.

Tools- Prolog

Task 2: Implementation of Hill climbing algorithm for Heuristic search approach using following constraints in python.

i) Create a function generating all neighbours of a solution
ii) Create a function calculating the length of a route
iii) Create a random solution generator
iv) Create a Travelling salesman problem

Tools- Python, Online Simulator - https://graphonline.ru/en/

Task 3: Implementation of A * Algorithm to find the optimal path using Python by following constraints.

- The goal of the A* algorithm is to find the shortest path from the starting point to the goal point as fast as possible.
- The full path cost (f) for each node is calculated as the distance to the starting node
- (g) plus the distance to the goal node (h).
- Distances is calculated as the manhattan distance (taxicab geometry) between nodes.

Tools- Python, Online Simulator - https://graphonline.ru/en/

Task 4: Implementation of Mini-Max algorithm uses recursion to search through the game-tree using python by applying following constraints.

- In this algorithm two players play the checker's game; one is called MAX and other is called MIN.
- Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
- Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
- The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
- The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

Tools : Python

Task 5: Implementation of Ant Colony Optimization to Optimize Ride-Sharing Trip Duration using Python by following constraints.

- To forecast travel times between every pair of pick-up and drop-off locations.
- To find the shortest route that visits a set of locations.
- To implement optimization techniques are required to intelligently search the solution space and find near-optimal solutions.

Tools: Python

Task 6: Solve a Map Coloring problem using constraint satisfaction approach by applying following constraints

- Assign each territory a color such that no two adjacent territories have the same color by considering following parameters: Domains, Variables and Constraints
- Apply Basic Greedy Coloring Algorithm: Color first vertex with first color, do following for remaining V-1 vertices.
- Consider the currently picked vertex and color it with the lowest numbered color that has not been used on any previously colored vertices adjacent to it. If all previously used colors appear on vertices adjacent to v, assign a new color to it.

Tools- Python, Online Simulator - https://graphonline.ru/en/

Task 7: Implementation of Monkey Banana Problem in Goal Stack planning using prolog by applying following constraints.

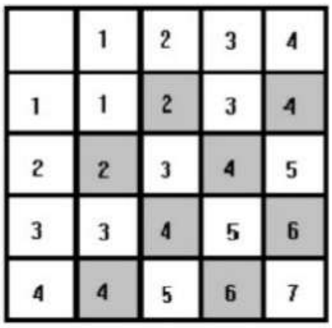
Imagine a room containing a monkey, chair and some bananas. That have been hanged from the centre of ceiling. If the monkey is clever enough, he can reach the bananas by placing then chair directly below the bananas and climb on the chair. The problem is to prove the monkeycan reach the bananas.The monkey wants it, but cannot jump high enough from the floor. At the window of the room there is a box that the monkey can use. The monkey can perform the

Following actions: -

1) Walk on the floor.
2) Climb the box.
3) Push the box around (if it is beside the box).
4) Grasp the banana if it is standing on the box directly under the banana.

Tools: Prolog

Task 8: Implementation of N-queen problem using backtracking algorithm using prolog In the 4 Queens problem the object is to place 4 queens on a chessboard in such a way that no queens can capture a piece. This means that no two queens may be placed on the same row, column, or diagonal.



	1	2	3	4
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6
4	4	5	6	7

The n Queens Chessboard.

Tools: Prolog

Task 9: To Build an Intelligent Chatbot system with Python and Dialog-flow using Interactive Text Mining Framework for Exploration of Semantic Flows in Large Corpus of Text.

- To integrate with Google Cloud Speech-to-Text and third-party services such as Google Assistant, Amazon Alexa, and Facebook Messenger.
- Configure Dialogflow to manage your data across GCP services and let you optionally integrate Google Assistant.

Tools- Python, Dialog-flow Framework

Task 10: Implement simple fact for following:

a. Ram likes mango.
b. Seema is a girl.
c. Bill likes Cindy.
d. Rose is red.
e. John owns gold.

Tools-Prolog

PART-2

Use Cases:

- Implementation of Map navigation using Heuristic search approach in python.
- Develop a simple neural network for simulating logic gate operations.
- Solve Multiple criteria production scheduling problem using Genetic algorithm
- Implementation of Text Lemmatization using NLTK Python Package.
- Implementation of Intelligent Chat-Bot system using Python and Dialogflow
- Implementation of Block world problem using STRIPS domain precondition rules in prolog.

Total: 30 Hours(P)

H. Learning Resources

i. Text Book:

1. Stuart. J. Russell, et al. "Artificial Intelligence by Pearson: A Modern Approach" 4th Ed, 2020.

ii. Reference Books:

1. E.Rich and K. Knight," Artificial Intelligence", Mc Graw Hill Publishers Inc, 3rd Edition, 2017.
2. Deepak Khemani, "A First Course in Artificial Intelligence", McGraw Hill Education, 2013.

ii. Online References:

1. "Artificial Intelligence: Search Methods for Problem Solving", May. 06, 2014 [Online]. Available: https://nptel.ac.in/courses/106/106/106106126/
2. "Artificial Intelligence: Knowledge representation and reasoning" Jan. 19, 2016 [online] Available : https://nptel.ac.in/courses/106/106/106106140/

COURSE CODE	COURSE TITLE	L	T	P	C
10212CC225	Ethical Hacking	3	0	2	4

A.Preamble

This course enables to provide the skills in hacking techniques, methodologies, tools, tricks and security measures to secure an organization's IT systems. Starting from basics of networking, network security, and cryptography, the course will cover various attacks, vulnerabilities, and ways to secure them. There will be hands-on demonstrations that will implement the security frameworks including preventive and detective controls which be helpful to the students.

B.Prerequisite Course

10211CC130–Fundamentals of Computer Networks

C.Course Objectives

Students undergoing this course are expected to

- Understand the concepts of ethical hacking with terminologies.
- Familiarize the tools to help competitive intelligence, DNS zone, and port scanning.
- Mitigate enumeration in the system and its vulnerabilities.
- Understand how to hack the server and applications, procure knowledge of attacks, and standards in technology.
- Analyze different attacks, and avalanche analysis viz IDS and NPS.

D.Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Identify and classifyattacks and threat vectors.	**K2**
CO2	Execute reconnaissance on a target network using a variety of scanning and probing techniques.	**K3**
CO3	Explore Microsoft and Linux Operating Systems, Network vulnerabilities.	**K3**
CO4	Demonstrate various web application vulnerabilities and wireless network attacks.	**K3**
CO5	Operate cryptography and hashing methods, Intrusion Detection and Prevention System.	**K3**
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E.Correlation of COs with Program outcomes and Programme Specific Outcomes:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	2	3		3	3	3	2						
CO2	2	1	1	1	3	1	1						2
CO3	3	2	3	3	3	1							3
CO4	3	1		1	3		1						3
CO5	3		2		3								3

3- High; 2-Medium; 1-Low

F.Course Contents

Unit 1 Ethical Hacking Overview L-9 Hours

Introduction to Ethical Hacking – Types of hackers – Attack Types, and Vectors – Threat categories – Malicious Software - Protecting against malware attacks – Intruder Attacks – Physical Security.

Unit 2 Reconnaissance and Scanning L-9 Hours

Introduction to Reconnaissance, Types of Reconnaissance - Footprinting – Web Tools – DNS Zone Transfers – Social Engineering – Port Scanning – Types of Port Scanning – Port Scanning tools – Ping Sweeps – Scripting.

Unit 3 Enumeration L-9 Hours

Introduction to EnumerationTools and Techniques – Windows, Unix Operating System – NetBIOS – NetBIOS Null Session, NetBIOS Enumeration Tools– Windows OS Vulnerabilities – Hardening Windows OS- Linux OS Vulnerabilities, Network Vulnerabilities.

Unit 4 Web Server and Wireless Hacking L-9 Hours

Understanding Web Application – Web Application vulnerabilities –OWASP - Web Application Injection Attacks - Code injection, SQL injection, cross-site scripting -Tools for Web Attackers and Security Testers – Wireless Technology – Wireless network standard – Authentication – Wardriving – Wireless Hacking

Unit 5 Cryptography Attacks and Network Protection Systems L-9Hours

Basics of Cryptography attacks – Brute Force Attacks, Replay Attack, Frequency Analysis and the Ciphertext Only Attack, Network Protection systems: Routers – Firewalls – Intrusion Detection and Prevention System - Honey pots.

Total: 45 Hours

G.Laboratory Experiments

PART - 1

Task 1: Installation of VM work station and Kali Linux/Parrot Operating System.

Task 2: Scan and identify the network range, different types of ports available on different networks, and Operating systems used by the target system/network.

Tool: Nmap

Task 3: Test the different web applications on the internet and write about security issues found for each website. Demonstration of man-in-the-middle attack.

Tool: Burpsuite

Task4: Perform the vulnerability scan and identify the different types of vulnerabilities present in the network.

Tool: OpenVAS

Task5: Perform vulnerability Scan, Exploitation, and Hardening on Windows/Linux OS using Shell Script/C/Python Program.

Task6: Perform code injection attacks on any Database and take over it.

Tools: SQL Map

Task7: Gain access to the WIFI router without having a password and modify configurations.

Tools: Aircrack ng, Airsnort, Ettercap

Task8: Configure Windows and Linux firewalls/Snort and manage user permissions by writing firewall rules.

Tools: Windows firewall, Snort

Task9: Install and Configure IDS system and identify the attacks performed on it. Mitigate the attacks by modifying the rules.

Tools: Snort, Suricata, Sagan

Task10: Configure, Perform, and Detect attacks on Honeypot.

Tool: Cowire

PART-2

Use Cases:

Use Case 1:

On April 11th 2011, at nine in the evening, Barracuda Networks posted a grim entry on their blog. Their network had been hacked. Thousands of their confidential customer and employee records were stolen. In an ironic twist of fate, the company that advocates security through its own Web Application Firewall were victims to the most common and oldest type of attack against web servers. Explain the different tools used by the Hacker. To analyze the effect that different training regimens have on player performance

Use Case 2:

A while back, the IT Help Desk received several complaints that one of the employee’s computers was sending out spam. They checked it out, band the reports were true: a hacker had installed a program on the computer that made it automatically send out tons of spam email without the computer owner’s knowledge. Explain the hacker’s technique to get into the computer to set this up.

Use Case 3:

Contactless technology was developed by Visa card and Mastercard with the mindset of never sacrificing security for convenience. The cards and devices contain an embedded chip and a radio frequency (RFID) antenna that provide a wireless link with the contactless reader. When the card or device is tapped against the reader, information is transmitted in a highly secure manner within a fraction of a second. Investigate how card payment frauds are occurring in Retail and Online shopping.

Total: 30 Hours

H. Learning Resources

i. Text Books:

1. Michael T. Simpson, Kent Backman, James Corley, “Hands-On Ethical Hacking and Network Defense”, Cengage Learning, Second Edition, 2012. [Unit 1-3,5]
2. “Ethical Hacking and Countermeasures”, EC-Council, Publisher: Course Technology; 2nd edition, 2016. [Unit 4]

ii. Reference Books:

1. Allen Harper, Shon Harris, Jonathan Ness, Chris Eagle, “Gray Hat Hacking The Ethical Hackers Handbook”, McGraw Hill Education, Third Edition, 2017.
2. Sean Philip Oriyano and Michael Gregg, “Hacker Techniques, Tools and Incident Handling”.
3. Jon Ericson, “Hacking: The Art of Exploitation”, Second Edition, No Starch Press, 2008, ISBN 978-1593271442.
4. Michael Sikorski, Andrew Honig, “Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software”, No Starch Press, 2012.

iii. Online References:

1. Grey Campus - “Ethical Hacking”. Accessed on: April, 20, 2021 [Online]. Available: https://www.greycampus.com/opencampus/ethical-hacking

COURSE CODE	COURSE TITLE	L	T	P	C
10212CC226	Secure Coding	3	0	2	4

A. Preamble

Secure coding is the practice of developing computer software in a way that guards against the accidental introduction of security vulnerabilities. This course comprises the concepts of various phases in secure software development life cycle, identifying the common programming error that result in vulnerabilities, and mitigating those errors. This course is intended for making applications more resistant to threats by applying various secure coding principles.

B. Prerequisite Courses

10210CS101 - Problem Solving using C

C. Course Objectives

- To impart knowledge on secure software development life cycle and security principles for building resilient applications.
- To identify common programming errors and vulnerabilities in C, C++, Java, and Python, and apply secure coding practices to mitigate them.
- To develop skills in threat modeling, risk mitigation, and security testing to build and deploy secure software systems.

Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Understand the concepts of various phases in secure software development life cycle.	K2
CO2	Explain the modeling of threats and mitigating those threats.	K2
CO3	Determine the common programming errors that result in software vulnerabilities and reducing those errors using C and C++ programming.	K3
CO4	Identify the software vulnerabilities and mitigating them using Java and Python programming.	K3
CO5	Apply testing for making applications more resistant to security threats and perform secure software installation	K3
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of COs with Program outcomes and Programme Specific Outcomes:

Cos	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	2												
CO2	2												
CO3	2				3	2							2
CO4	2				3	2							2
CO5	2				3	2							2

3-High; 2-Medium; 1-Low

F. Course Contents

Unit 1 Proactive Security Development Process **L-9 Hours**

Basics of Attacks, Security and Vulnerabilities - Need for secure systems - Secure Software Development Cycle (S-SDLC) - Security issues while writing SRS - Design phase security - Development Phase - Test Phase- Maintenance Phase – Best Practices SD3 (Secure by design, default and deployment) - Security principles.

Unit 2 Threat Modelling Process **L-9 Hours**

Identifying the Threats by Using Attack Trees and rating threats using DREAD - Risk Mitigation Techniques and Security Best Practices- Security techniques: authentication – authorization - Defence in Depth and Principle of Least Privilege.

UNIT 3 - Secure Coding in C and C++ **L-9 Hours**

Character strings- String manipulation errors – String Vulnerabilities and exploits – Mitigation strategies for strings- Pointers – Mitigation strategies in pointer based vulnerabilities – Dynamic memory management- Common C++ memory management errors - Memory managers

UNIT 4 - Secure Coding in Java and Python **L-9 Hours**

Java secure coding practices – overview – Input validation and output encoding- secure database queries – Errors and exceptions in Java – Security best practices for Python Programming.

UNIT 5 Testing Secure Applications **L-9 Hours**

Security Testing: The Role of the Security Tester - Building the Security Test Plan - Testing Clients with Rogue Servers - Testing with security templates - Test the end to end solution Determining Attack surface - Performing security code overview - Secure software installation - Introduction to full stack developer.

Total : 45 Hours (L)

Laboratory Experiments

Part - 1

TASK 1	Study on general secure coding practices
TASK 2	Demonstration of handling input validation
TASK 3	Implementation of safe coding by handling string management.
TASK 4	Implementation of safe coding by handling issues with pointers
TASK 5	Implementation of handling program output (Output Encoding)
TASK 6	Demonstration of authentication and password management (includes secure handling of credentials by external services/scripts)
TASK 7	Demonstration of secure Session Management in web applications.
TASK 8	Implementation of access controls in software.
TASK 9	Demonstration of Unvalidated File Upload vulnerability
TASK 10	Demonstration of handling errors and its logging
TASK 11	Demonstrate the buffer overruns in C language.
TASK 12	Demonstration of Damn Vulnerable Web Application (DVWA) kit for secure coding and testing.

TASK 13	Demonstration of Integrated DevSecOps Security Assessment using any one AI tool Rapid7, Qwiet AI, BlockDojo, SonarQube, and GitHub Advanced Security for vulnerability analysis and secure code validation.

Part-2

Use Cases:

- In the healthcare management system, protection of privacy and security of patient is a responsibility of data holding organizations. With medical data these concerns assume higher pronounce and priority. Medical data can be very sensitive, and may be linked to life and death of the patients.
- In the railway online reservation system, the passenger needs to create an account entering all personal information (Name, Address, Age, Sex, Contacts etc.), thereby entering journey details and making necessary payment. On the payment confirmation, the ticket details are received by the passenger (in SMS/E-mail etc.). The passenger produces this SMS/Printout of E-ticket along with Identity card to the Ticket Examiner during Journey.
- Secure Banking System: Information assets to be protected in the banking industry. As modern banking increasingly relies on the internet and computer technologies to operate their businesses and market interactions, the threats and security breaches are highly increase in recent years. Insider and outsider attacks have caused global businesses lost trillions of Dollars a year. It is vital to govern the information security in banking system.

Total: 30 Hours (P)

H. Learning Resources

i. Text Book:

1. Michael Howard, David LeBlanc, “Writing Secure Code”, Microsoft Press, Second Edition, Revision 2014. (UNIT-1, UNIT-2, UNIT-5)
2. Robert C.Seacord, “Secure Coding in C and C++”, Pearson Education, Second edition, 2013. (UNIT-3,UNIT-4)

ii. Reference Books:

5. Frank Swiderski, Window Snyder, “Threat Modeling”, Microsoft Professional, First Edition, 2004.
6. Bhargav A, Kumar BV, “Secure Java: for web application development”, CRC Press, 2010.
7. Tamaghna Basu, “Secure Programming with Python”, Packt Publishing Limited, 2017.
8. Julia H. Allen, Sean J. Barnum, Robert J. Ellison, Gary McGraw, Nancy R. Mead, “Software Security Engineering: A guide for Project Managers”, Addison-Wesley Professional, 2008.

iii. Online References:

1. “Secure Coding with Python” Accessed on: July. 07, 2021 [Online]. Available: https://devopedia.org/secure-coding-with-python
2. Secure Coding in Python Accessed on: July. 07, 2021 [Online]. Available: https://codehandbook.org/secure-coding-in-python/
3. “Secure Programming HOWTO ”, Accessed on: July. 01, 2021 [Online]. Available: https://dwheeler.com/secure-programs/Secure-Programs-HOWTO/
4. “Java Coding Guidelines”, Accessed on: July. 01, 2021 [Online]. Available: https://wiki.sei.cmu.edu/confluence/display/java/Java+Coding+Guidelines
5. “Secure Coding Practice Guidelines”, Accessed on: June. 30, 2021 [Online]. Available: https://security.berkeley.edu/secure-coding-practice-guidelines
6. “Secure Coding - Secure application development”, https://www.udemy.com/course/secure-coding-secure-application-development/

COURSE CODE	COURSE TITLE	L	T	P	C
10212CC228	Blockchain Technology	2	0	2	3

A. Preamble

Blockchain technology has become popularly known because of its use in the implementation of Cryptocurrencies such as BitCoin, Ethereum. The technology itself holds much more promise in various areas such as time stamping, logging of critical events in a system, recording of transactions and trustworthy e-governance. This course covers the technical aspects of public distributed ledgers, blockchain systems, cryptocurrencies, and smart contracts. Students will learn how these systems are built, how to interact with them, how to design and build secure distributed applications.

B. Prerequisite Courses

10211CC130 - Fundamentals of Computer Networks

C. Course Objectives

Learners are exposed to

- Assess blockchain applications in a structured manner.
- Impart knowledge in block chain techniques and able to present the concepts clearly and structured. To get familiarity with future currencies and to create own crypto token.

D. Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Explain Blockchain with types of consensus mechanisms and protocol	K2
CO2	Illustrate the concepts of Cryptocurrency and Smart Contracts to build blockchain	K3
CO3	Construct Public, Private and Consortium Blockchain System for real time implementation	K3
CO4	Use Initial Coin Offering, security and privacy in blockchain with Hyperledger Fabric	K3
CO5	Implement Blockchain applications in real time using solidity.	K3
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of COs with Program outcomes and Programme Specific Outcomes:

Cos	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	2	2							2	2
CO2	2	2	2	2	2							2	3
CO3	2	2	2	2	2							2	3
CO4	2	2	2	2	2							2	3
CO5	2	2	1	1	1							2	3

3-High; 2-Medium; 1-Low

F. Course Contents

Unit 1 Introduction to Blockchain, Types and Consensus Mechanism L-6 Hours

Fundamentals of Blockchain- Introduction - Origin of Blockchain - Blockchain Solution - Components of Blockchain - Block in a Blockchain - Technology and the Future.

Blockchain Types and Consensus Mechanism - Introduction - Decentralization and Distribution -Types of Blockchain - Consensus Protocol.

Unit 2 Cryptocurrency and Smart Contracts L-6 Hours

Cryptocurrency – Bitcoin, Altcoin and Token – Bitcoin and the Cryptocurrency - Cryptocurrency Basics - Types of Cryptocurrencies - Cryptocurrency Usage.

Smart Contracts - Characteristics of a Smart Contract - Types of Smart Contracts -Types of Oracles - Smart Contracts in Ethereum - Smart Contracts in Industry.

Unit 3 Types of Blockchain L-6 Hours

Public Blockchain System - Popular Public Blockchain - Bitcoin Blockchain -Ethereum Blockchain.

Private Blockchain System - Key Characteristics of Private Blockchain – Need of Private Blockchain -Private Blockchain Examples- Private Blockchain and Open Source - Various Commands (Instructions) in E-commerce Blockchain -Smart Contract in Private Environment - State Machine -Different Algorithms of Permissioned Blockchain -Byzantine Fault – Multichain. **Consortium Blockchain** - Key Characteristics of Consortium Blockchain -Need of Consortium Blockchain - Hyperledger Platform -Overview of Ripple - Overview of Corda.

Unit 4 Security and Application of Blockchain L-6 Hours

Initial Coin Offering - Blockchain Fundraising Methods -Launching an ICO - Investing in an ICO - Pros and Cons of Initial Coin Offering - Successful Initial Coin Offerings - Evolution of ICO- ICO Platforms.

Security in Blockchain - Security Aspects in Bitcoin -Security and Privacy Challenges of Blockchain in General - Performance and Scalability -Identity Management and Authentication Regulatory Compliance and Assurance -Safeguarding Blockchain Smart Contract (DApp) - Security Aspects in Hyperledger Fabric.

Unit 5: Development Tools and Frameworks L-6 Hours

Blockchain Ethereum Platform using Solidity - Remix IDE -Structure of a Smart Contract Program -Using Remix to Write and Run a Solidity Program – Modifiers – Events - Arrays in Solidity - Function Visibility - Variable Visibility - Function Modifier Keyword -How Funds Are Accepted - Fallback Function - Contract Inheritance -Contract Communicating with Another Contract - External Libraries - ERC20 Token Transfer - Error Handling in Solidity - Application Binary Interface (ABI) - Swarm (Decentralized Storage Platform) -Whisper (Decentralized Messaging Platform)

objects- relations – events- actions- time and space-reasoning with defaults- Reasoning about knowledge Case studies – chatbot system- recommendation system.

Total: 30 Hours(L)

G. Laboratory Experiments

PART – 1

TASK 1	Creating MetaMask wallet and crypto ● Meta Mask Setup ● Deposit the Crypto ● Make a Transaction and Extracting the transaction details ● Ethereum Transaction **Tool:** MetaMask
TASK 2	Implement the Smart Contract using Remix, Solidity ● Working With Variables ● Sending Money ● Starting, Pausing, Stopping and Deleting transaction ● Mapping and Struct ● Error Handling ● View/Pure, Receive Function and Fallback Function ● Inheritance, Modifier and Importing ● Events and Return Variables **Tool:** Remix, Solidity
TASK 3	Installation and Configuration of Node.js and Web3.js ● Transfer Ether ● Interacting with Smart Contract ● Using Web3.js with Chrome to Interact with Smart Contracts **Tool:** Node.js and Web3.js
TASK 4	Creating the Hyperledger Fabric using solidity ● Create the Fabric Test Network ● ringing up the test network ● Creating a channel ● Starting a chaincode on the channel ● Interacting with the network ● Bringing down the network **Tool:** Node.js and Web3.js
TASK 5	Creating the Shared Wallet using solidity
TASK 6	Deploying MEW with Ganache
TASK 7	Establish the transaction using Remix with Ganache **Tool:** Remix with Ganache

PART-2

Use Cases:

Use Case 1: Retail - Establishing unconditional transparency in the food supply chain using blockchain Hyperledger fabric.

Use Case 2: Banking and Financial Services – Increasing transaction volume and network resilience while maintaining confidentiality requirements for real time gross settlement among different banks of a central banking consortium.

Use Case 3: Healthcare - Electronic health records and medical research data digitization.

Use Case 4: Energy and Utilities – Renewable energy trading which are locally owned deliver benefits that are environmentally relevant for the communities involved.

Use Case 5: Blockchain in Real-estate – The real estate market currently faces several issues related to housing affordability, rising rate and economy, many of which cannot be addressed with blockchain.

Total: 30 Hours(P)

H. Learning Resources

i. Text Book:

1. Chandramouli Subramanian, Asha A George, Abhilash K A and MeenaKarthikeyan, "Blockchain Technology", Universities Press,2021(Unit1-Unit5)
2. Imran Bashir, "Mastering Blockchain: Distributed ledger technology, decentralization, and smart contracts explained", Birmingham – Mumbai: Packt, 2nd Edition Kindle, 2018 (Unit 1&2)

ii. Reference Books:

1. Narayanan, Arvind, et al. Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction. Princeton University Press, 2016.
2. Antonopoulos, Andreas M. Mastering Bitcoin: Programming the Open Blockchain. O'Reilly Media, Inc., 2017.
3. Antonopoulos, Andreas M. and Wood, Gavin. Mastering Ethereum. O'Reilly Media, Inc., 2018.

iv. Online References:

1. Ethereum project documentation", Jan 21 2016, Accessed on: Jan 21 2016 [Online]. Available: https://ethdocs.org/en/latest/
2. "Monero Research Lab papers" 01 April2014, Accessed on: 01 April 2014 [Online]. Available: https://www.getmonero.org/resources/research-lab/
3. "ZCash documentation", 10 Jan 2021 Accessed on: 10 Jan 2021 [Online]. Available: https://z.cash/

COURSE CODE	COURSE TITLE	L	T	P	C
10212CC230	Forensics in Cyber Security	3	0	2	4

B. Preamble

This course will address methods to properly conduct a computer forensics investigation. It includes digital evidence collection and evaluation of network and host system intrusions with hands-on use of powerful forensic analysis tools.

B. Prerequisite Course

10211CC130 – Fundamentals of Computer Networks
10211CC103 – Operating Systems

C. Course Objectives

Learners are exposed to

- Introduction to Digital Forensics and Evidences
- Network Forensics
- Forensic Approaches and Forensic Duplication
- Digital Forensic Tools
- Incident Response

D. Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K – Level
CO1	Outline and Explore the fundamentals of cyber forensic analysis.	K2
CO2	Infer and conduct basic network forensic analysis	K3
CO3	Classify the evidence acquisition and forensics duplication	K2
CO4	Evaluate and validate the types of Forensic tools	K2
CO5	Detect cyber security incidents and its response	K2
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of COs with Program outcomes and Programme Specific Outcomes:

Cos	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	2	1		1		2					2		
CO2	1	1	1	1	1	3					2		
CO3	1	2				3				2		2	
CO4	2		2		2	2				1			2
CO5		3		3	3	3	2						

3-High; 2-Medium; 1-Low

F. Course Contents

Unit 1 Forensic Science **L-9 Hours**

Principles and Methods: Scientific approach to Forensics, Identification and Classification of Evidence, Location of Evidence: Storage Media, Hard drives, Hardware Interfaces, Recovering Data, Media File Forensic Steps –Forensic Analysis: Planning, Case Notes and Reports, Quality Control.

Unit 2 Network Forensic **L-9 Hours**

Network forensics overview-Securing a Network-Developing procedures for network forensics-Investigating virtual networks-Examining Honey net projects-E-mail Investigations: Role of client and server in E-mail, Investigating E-mail crimes and violations, E-mail Servers, E-mail Forensic tools.

Unit 3 Digital Forensic Approaches and Forensic Duplication **L-9 Hours**

Introduction-Policy and Procedure Development – Evidence Assessment Evidence Acquisition – Evidence Examination - Documenting and Reporting- Introduction to Forensic Duplication - Rules of Forensic Duplication (Thumb Rule) - Necessity of Forensic Duplication - Forensic Duplicates as Admissible Evidence - Important Terms in Forensic Duplicate - Forensic Duplication Tool Requirements - Creating a Forensic Duplicate of a Hard Drive.

Unit 4 Forensics Tools **L-9 Hours**

Evaluating Forensics Tool Needs – Tasks performed by forensics tools –Forensics Software Tools: Command – line forensic tools, Linux forensic tools –Forensics Hard ware Tools: Forensic workstation, Write-Blocker-Validating and Testing Forensics Software.

Unit 5 Incident Response **L-9 Hours**

Incidence Response Goals of Incident Response – People Involved in Incident Response Process - Incident Response Methodology - Activities in Initial Response - Phases after Detection of an Incident – Report Writing and Presentation.

Total: 45 Hours (L)

G. Laboratory Experiments

<u>PART – 1</u>

Task 1: Study of Computer Forensics and different tools used for forensic investigation

Task 2: To extract Exchangeable image file format (EXIF) Data from Image Files.
Tool-Exifreader Software

Task 3: Forensics Case Investigation using live data
Tool-Autopsy

Task 4: To Recover Deleted Files using Forensics Tools
Tool-FTK/File Analyzer

Task 5: To Find Last Connected USB on your system
Tool-Parse

Task 6: To View Last Activities on Your PC
Tool-Register Editor/Registry Explorer

Task 7: To analyze network-related incident
Tool-Wireshark

Task 8: To make the forensic image of the hard drive to find the suspicious activity
Tool-EnCase

Task 9: To Restore the Evidence Image to find criminal activity
Tool-EnCase

Task 10: To hide and extract any text file behind an image file/Audio file using Command Prompt. **Shell comments/ Shell Bags Explorer**

Task 11: To capture the physical memory of a computer and analyze artifacts in memory
Tool-Magnet RAM Capture

Task 12: To Collect Email Evidence in Victim PC
Tool-Email Analyzer

PART-2

Use cases 1:
You are on-site, conducting a preliminary examination of a Linux system. The hardware suite includes a 56KB modem. What areas of search should be included in your examination? Prepare an examination plan that details what you will look for, and why.

Use cases 2:
You receive a Windows OS X system and are asked to summarize the applications and data on the hard drive. In addition, you are asked to report any recent system usage and any signs of encryption, external storage media, or clock tampering.

Use cases 3:
As a security investigator, you have been asked to determine if company confidential information (intellectual property) has been copied from enterprise computers. Investigation centers on a particular computer that has shown a high volume of network traffic at unusual times. In the course of conducting your investigation you discover that large capacity removable media has been attached to the suspect computer. A preview examination reveals that software used for secure deletion had been downloaded to the desktop. Prepare an investigative plan listing the lines of investigation that you plan to pursue.

Use cases4:
A large accounting MNC company is going to audit certain activities by officers of a medium size, publicly traded bank. During the investigation, the appointed auditor needed to examine several computer systems used by certain Bank employees. After collecting the evidence, the digital forensic examiners were immediately dispatched and sent in to arrange for the formal investigation of those systems to search for corroborating evidence in support of the audit team’s suspicions and findings.

Use cases5:
Discover you have known the existence of threatening e-mails being sent to the CEO of your company at that time predict will you do the examination of the e-mails revealed that they originated from outside of the country? Describe the steps you would take in your investigation. In particular, address the issue of jurisdiction and locating your counter parts in the target country.

Total: 30 Hours (P)

H. Learning Resources

i. Text Book:

1. Chuck Eastom,"Certified Cyber Forensics Professional Certification, McGraw Hill, July 2017. (Unit1)
2. Bill Nelson, Amelia Philips and Chris Steuart, "Guide to Computer Forensics and Investigations",Cengage Learning, 6thEdition, 2019(Unit2)
3. Greg Gogolin,"Digital Forensics Explained", CRC Press Taylor and Francis Group, 2013 (Unit3,5)
4. Nilakshi Jain and Dhananjay R Kalbande, "Digital Forensic: The Fascinating World of Digital Evidences", Wiley Press, 2017 (Unit4)

ii. Reference Books:

1. Nhien-AnLe-Khac, "Security, Privacy, and Digital Forensics in the Cloud" Wiley Press, 2019, ISBN-13: 978-1119053286ISBN-10:1119053285.
2. Nihad A. Hassan "Digital Forensics Basics A Practical Guide Using Windows OS", A press Publishers, 2019.
3. Mike Sheward "Digital Forensic Diaries" Secure owl, 2017, ISBN: 9781521514467.
4. Gerard Johansen, "Digital Forensics and Incident Response A practical guide to deploying digital forensic techniques in response to cyber security incidents".

iii. Online References:

1. https://online.norwich.edu/academic-programs/resources/5-steps-for-conducting-computer-forensics-investigations
2. http://resources.infosecinstitute.com/computer-forensics-tools
3. http://www.cybrary.it/course/computer-hacking-forensics-analyst

COURSE CODE	COURSE TITLE	L	T	P	C
10212CC232	Vulnerability Management and Penetration Testing	3	0	2	4

A. Preamble

This course will enable the students to understand and apply the underlying principles and many of the techniques associated with the cyber security practice that starts with introduction to vulnerability analysis and continues further to explore the concepts such as penetration testing, the exploitation in the networks and host through vulnerability analysis. This course covers vulnerability management process from passive and active reconnaissance, vulnerability assessment, post exploitation and finally vulnerability detection and avoidance using fuzzing, string and taint analysis concepts.

B. Prerequisite Courses

10211CC219 – Cyber Security

C. Course Objectives

Learners are exposed to

1. Interpret the vulnerabilities and exploitation in the system
2. Identify the cyber-attacks and their possible remedial measures undertaken
3. Interpret a management view of system and apply security strategy through efficient vulnerability analysis and cyber security principles.
4. Infer the security threats in context of users, organization and third party risk assessment.
5. Make use of Vulnerability scanning tools to mitigate the cybersecurity incidents

D. Course Outcomes

Upon the successful completion of the course, students will be able to:

CO No's	Course Outcomes	K - Level
CO1	Identify the basic concepts of vulnerabilities, their terminologies and overview of testing methods	K3
CO2	Apply the penetration testing tools to explore the vulnerabilities in a system	K3
CO3	Organize the management of vulnerabilities including discovery and mitigation	K3
CO4	Identify the exploitation and the possible threats through various attacking mechanisms	K3
CO5	Make use of the vulnerability and scanning tools to detect and avoid vulnerabilities	K3
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of COs with Program outcomes and Programme Specific Outcomes:

Cos	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	2	3	3	2	3	3	2			3		3	3
CO2	1	3	2	1	3	3	1		1	2		3	3
CO3	2	3	1	1	3	2	2			1		1	3
CO4	2	3	1	2	3	3	1		1			1	3
CO5	1	3	3	1	3				1			2	3

3-High; 2-Medium; 1-Low

F. Course Contents

Unit 1 Introduction to Vulnerability Analysis L-9 Hours

Introduction to vulnerability – Types of Vulnerabilities - Vulnerability assessment (VA) – Security weakness in a system –Host level Vulnerability – Network level vulnerability – Application level vulnerability – Reconnaissance – Active and Passive Reconnaissance – Vulnerability testing – Testing Methodologies.

Unit 2 Penetration Testing L-9 Hours

Introduction to Penetration testing - An Overview of Ethical requirements and legal issues - Penetration test report and components – Scanning network - DNS, web reconnaissance – Patching - Buffer overflow - Introduction to Nessus scanning tool – Introduction to Metasploit framework - Metasploit security vulnerabilities.

Unit 3 Vulnerability Management L-9 Hours

Introduction – Information gathering – Active and Passive Information gathering – Scanning for Vulnerabilities - Who is lookup – Introduction to vulnerability management – Vulnerability discovery – Vulnerability Mitigation – Patch and Configuration management – Encountering DoS and DDoS attack – Web application vulnerability scanning – Arachni tool.

Unit 4 Vulnerability and Exploitation L-9 Hours

Introduction - Cyber ethics – Cyberattack – SQL Map – Broken Authentication – Sensitive Data exposure – XML external entity attack – Code Injection – CRLF injection - Broken access control – Security misconfigurations – Cross site Scripting – Deserialization attack – Using components with known vulnerabilities – Logging and auditing dashboard.

Unit 5 Vulnerability Detection and Avoidance L-9 Hours

White box and Black box testing in cybersecurity - Introduction to white box, black box and grey box fuzzing –- Taint analysis – String analysis and solving – Protection from format string vulnerabilities – Protection from buffer overflow vulnerabilities – Network analysis tools – Port scanning and use of scripts in nmap script engine – Report Writing – Post incident log review.

Total: 45 Hours (L)

G. Laboratory Experiments

PART – 1

TASK 1	Understanding kali linux and retrieving Personal identifiable information of a host such as registrant name, organization, country, name servers, and date of creation, expiry and updation **Tools: Kali Linux**
TASK 2	Perform a web application security scan to identify vulnerabilities by arachni tool using Kali linux **Tools: Kali Linux, Arachni**
TASK 3	To identify the vulnerable hosts using SQLiv for performing SQL injection attack using SQL map in Kali Linux **Tools: SQLmap, Kali Linux**
TASK 4	To detect, exploit and report cross-site scripting in web based applications using Xsser in kali linux **Tools: Xsser, Kali Linux**
TASK 5	To perform analysis on the list of infected files downloaded and their hashes, and analyse the URL/Domain of the infected site using Wireshark **Tools:Wireshark**
TASK 6	To sniff packets using SSH and Telnet protocol and retrieve the username and password using Wireshark **Tools: Wireshark**
TASK 7	Introduction to operations on Metasploit framework and to identify vulnerabilities and information gathering in a target system **Tools: Metasploit**
TASK 8	To perform the privilege Escalation attack scenario by identifying the exploit and payload using metasploit **Tools: Metaspolit**
TASK 9	To engage target systems to gather information about vulnerabilities and perform the scanning operations for services in which each host is operating in the target using nmap **Tools: Nmap**
TASK 10	Use Nmap scripting engine (NSE) to modify the predefined nmap scripts to identify customized information gathering or vulnerabilities.**Tools: Nmap**

PART-2

Use Cases:

Use Case 1: Scan an http domain site and assess its vulnerabilities to prepare a detailed vulnerability report including Assessment overview, individual vulnerability details and prioritizing the vulnerabilities.
Use Case 2: Perform a Penetration test on a Web application and detect its vulnerabilities associated and also remediate a solution for the vulnerability
Use Case 3: Analyze the security incidents in a web application and prepare a log and audit table providing the security incidents on a web application.
Use Case 4: Perform a scanning process in network and preparing reports on reverse DNS and IP lookups, PII

Total: 30 Hours (P)

H. Learning Resources

i. Text Book:

1. Patrick Engebretson, “The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy”, ISBN: 978-1597496551, Syngress, 2013.(Unit I, IV)
2. Stuart McClure, Joel Scambray, George Kurtz, “Hacking Exposed 7: Network Security Secret & Solutions”, McGraw Hill, ISBN 978-0-07-178028-5, 2012.(Unit V)
3. Wil Allsopp, "Advanced Penetration Testing: Hacking the World's Most Secure Networks", Wiley Publications, ISBN: 978-1-119-36768-0, 2018**.** (Unit II, III)

ii. Reference Books:

1. Joakim Kavrestad, “Fundamentals of digital forensics, theory, methods, and real life applications”, Second Edition, ISBN 978-3-030-38954-3, Springer, 2020

iii. Online References:

1. “Ethical Hacking”, Accessed on: May 2022 [Online]. Available: https://nptel.ac.in/courses/106/105/106105217/
2. “Vulnerability Scanning”, Accessed on: June 2022 [Online]. Available: https://alison.com/topic/learn/92039/web-application-vulnerability-scanning
3. “Vulnerability scanning tool: Nessus”, Accessed on: June 2022 [Online]. Available: https://www.cs.cmu.edu/~dwendlan/personal/nessus.html
4. “Vulnerability Testing”, Accessed on May 2022 [Online]. Available: https://us-cert.cisa.gov/bsi/articles/best-practices/white-box-testing/white-box-testing#BandG

COURSE CODE	COURSE TITLE	L	T	P	C
10212CC292	**Applied Programming Skills**	**3**	**0**	**2**	**4**

A. Preamble

This course develops problem-solving and programming skills using Java. It focuses on Data Structures and Algorithms concepts, encouraging students to implement, test, and optimize solutions for real-world computational problems. Students will learn to design efficient programs using arrays, linked lists, stacks, queues, trees, graphs, and advanced algorithmic techniques.

B. Prerequisite Course – Nil

C. Course Objectives

Learners are exposed to:

- Understand and use linear data structures for efficient data processing.
- Apply stacks, queues, trees, heaps, and graphs to solve programming problems.
- Use greedy, divide-and-conquer, and dynamic programming techniques.
- Choose suitable data structures and algorithms to solve problems effectively.
- Handle large inputs and time-critical programming scenarios.

D. Course Outcomes

Upon the successful completion of the course, students will be able to:

COs	Course Outcomes	K Level
CO1	Use linear data structure techniques to process competitive programming inputs efficiently.	**K3**
CO2	Implement stack- and queue-based patterns to solve constrained sequence problems.	**K3**
CO3	Apply tree and heap structures to handle hierarchical data and priority-based computations.	**K3**
CO4	Utilize graph and set-based structures to analyze connectivity, traversal, and relationship-driven problems.	**K3**
CO5	Develop algorithmic solutions using programming strategies for optimization problems.	**K3**
Knowledge Level (Based on revised Bloom's Taxonomy) K1-Remember K2-Understand K3-Apply K4-Analyze K5-Evaluate K6-Create		

E. Correlation of COs with Program Outcomes and Programme Specific Outcomes

Cos	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	2	1	1	1	-	-	-	-		2	3
CO2	3	3	2	2	1	1	-	-	-	-		2	3
CO3	3	3	3	2	2	1	-	-	-	-		3	3
CO4	3	3	3	3	2	1	-	-	-	-		3	3
CO5	2	3	3	3	3	2	-	-	1	2	1	2	3

3- High; 2-Medium; 1-Low

F. Course Contents

Unit I: Linear Data Structures for Competitive Input Processing L - 9 Hours

Two Pointers – Sliding Window – Prefix Sum – Difference Array – Binary Search – Quick Sort and Merge Sort, Greedy pairing – Hashing for frequency/count – In-place array manipulation – Linked list fast/slow pointer – Merge two lists – Reverse nodes in k-group – Detect/remove cycle – Intersection of linked lists.

Unit II: Stack and Queue Based Competitive Problem Modeling L - 9 Hours

Monotonic Stack – Next Greater Element pattern – Valid parentheses variants – Min Stack design – Stack simulation problems – Queue using stacks – Sliding window maximum using deque – FIFO processing with queue – Circular queue simulation – Task scheduling with queue.

Unit III: Hierarchical and Priority-Oriented Data Structures L - 9 Hours

Tree – Tree Traversal – Print Birdseye View of a Tree - Path sum - Root-to-leaf problems – Diameter/height problems – Lowest Common Ancestor – Kth smallest/largest in Binary Search Tree – Heap - Top-K pattern – K-way merge using heap.

Unit IV: Network and Relationship-Driven Data Structures L - 9 Hours

Graph – Breadth First Search –Depth First Search – Connected components – Cycle detection (directed/undirected) – Topological sorting– Shortest path in unweighted graphs – Grid traversal (islands, flood fill) – Union Find (Disjoint Set Union) – HashSet patterns – Frequency map and sorting by keys/values.

Unit V: Algorithm Design Techniques L - 9 Hours

Divide and Conquer pattern – Merge sort for counting inversions – Quick select (Kth element) – Dynamic Programming – One Dimensional (house robber, climbing stairs) –Two Dimensional (grid paths) – Subset Dynamic Programming (0/1 knapsack style) – Greedy interval scheduling – Backtracking (subsets, permutations, combinations).

TOTAL: 45 Hours

G. Laboratory Experiments

Part A

Task 1: (Level: Easy)

Given an array of intervals, merge all overlapping intervals.

Approach / Explanation: Sort by start. Iterate, merging when current.start <= last.end.

Sample Test Case:

Sample Input:

```
4
1 3
2 6
8 10
15 18
```

Sample Output:

1 6
8 10
15 18

Task 2: (Level: Easy)

Insert a new interval into a list of non-overlapping, sorted intervals, merging if necessary.

Approach / Explanation: Add all intervals ending before new.start, merge overlaps, then append the rest.

Sample Test Case:

Sample Input:

3
1 3
6 9
12 16
4 10

Sample Output:

1 3
4 10
12 16

Task 3: (Level: Easy)

You are given an array arr[] of size n - 1 that contains distinct integers in the range from 1 to n (inclusive). This array represents a permutation of the integers from 1 to n with one element missing. Your task is to identify and return the missing element.

Examples:

Input: arr[] = [1, 2, 3, 5]

Output: 4

Explanation: All the numbers from 1 to 5 are present except 4.

Input: arr[] = [8, 2, 4, 5, 3, 7, 1]

Output: 6

Explanation: All the numbers from 1 to 8 are present except 6.

Input: arr[] = [1]

Output: 2

Explanation: Only 1 is present so the missing element is 2.

Constraints:

1 ≤ arr.size() ≤ 106

1 ≤ arr[i] ≤ arr.size() + 1

Task 4: (Level: Easy)

Write a function that finds and returns the Nth prime number. The number N will be passed as input to the function

constraint: 1<=N<=1000,where N is the position of the prime number

Sample Test Case:

Sample Input:

10

Sample Output:

29

Task 5: (Level: Medium)

Given the head of a singly linked list, the task is to remove a cycle if present. A cycle exists when a node's next pointer points back to a previous node, forming a loop. Internally, a variable “**pos**” denotes the index of the node where the cycle starts, but it is not passed as a parameter. The terminal will print true if a cycle is removed otherwise, it will print false.

Examples:

Sample Input 1: head = 1 -> 3 -> 4->1, pos = 0

Sample Output 1: true

Explanation: The linked list has a loop is present in the list, and it is removed.

Sample Input 2: head = 1 -> 8 -> 3 -> 4, pos = 0

Sample Output 2: false

Explanation:

The Linked list does not contains any loop.

Sample Input 3: head = 1 -> 2 -> 3 -> 4->1, pos = 0

Sample Output 3: true

Explanation: The linked list has a loop is present in the list, and it is removed.

Constraints:

1 ≤ size of linked list ≤ 10^5

Task 6: (Level: Hard)

Library Management System

Scenario: A city-library chain wants to digitise its inventory and borrower system.

Internal tasks:

1. Design and implement data structures for book inventory (arrays or linked lists) and borrower records.
2. Implement borrow/return operations using a queue of pending returns and stack for recent returns.
3. Implement search (linear & binary) for books by title/ISBN, and sort books by popularity (using quick sort or merge sort).

Task 7: (Level: Easy)

Given a string s containing '()[]{}', determine if it is valid.

Approach / Explanation: Use a stack; push openings and check matching closings.

Sample Test Case:

Sample Input1:

()[]{}

Sample Output2:

True

Sample Input2:

( ]

Sample Output2:

False

Task 8 : (Level: Easy)

You are developing a music streaming app that manages user playlists. Each playlist is a collection of songs stored in an array of integers, where each integer represents a unique song ID. To enhance the listening experience, the app allows users to rotate the playlist to the right by a given number of positions. Write a program to rotate the playlist and print the updated order of songs.

Input Format

- The first line contains an integer n, the number of songs in the playlist.
- The second line contains n space-separated integers representing the song IDs.
- The third line contains an integer k, the number of positions to rotate the playlist to the right.

Output Format

- Print the updated playlist after rotation as n space-separated integers.

Sample Test Case:

Sample Input:

```
5
1 2 3 4 5
2
```

Sample Output:

```
4 5 1 2 3
```

Task 9: (Level: Easy)

You are given a string S that contains lowercase and uppercase English letters along with brackets '(' and ')'. Your task is to:

1. Remove all brackets '(' and ')' from the string.
2. Reverse the resulting string.
3. Print the final reversed string.

Input Format:

- The first line contains a string S (1 ≤ |S| ≤ 10^5) consisting of lowercase and uppercase letters and brackets '(' and ')'.

Output Format:

- Print the final string after removing brackets and reversing.

Sample Test Case:

Sample Input:

```
a(bc)d
```

Sample Output:

```
Dcba
```

Task 10: (Level: Hard)

E-commerce Inventory & Sales System

Scenario: An online store’s backend needs to manage product catalogue, sort & search products, and track top sellers.

Internal tasks:

1. Use arrays/linked lists to model product items (CRUD: create, read, update, delete).
2. Sort product lists by price, sales count, rating using various sort algorithms (bubble,

selection, insertion, quick, merge).

3. Use Java Collections: HashSet for unique product IDs; TreeMap for product → price mapping; PriorityQueue for top-selling products.

Task 11: (Level: Easy)

Given an integer array nums and integer k, return the k most frequent elements.

Approach / Explanation: Count with hashmap; use bucket sort (frequency -> list of values) or a min-heap of size k.

Sample test Case:

Sample Input 1:

1 1 1 2 2 3

2

Sample Output 1: 1 2

Task 12: (Level: Easy)

Given two strings s and t, determine if t is an anagram of s.

Approach / Explanation: Count characters with fixed-size array.

Sample Test Cases:

Sample Input 1:

listen

silent

Sample Output 1:

true

Sample Input 2:

Rat

tar

Sample Output 2:

False

Task 13: (Level: Medium)

You are given a binary tree with N leaf nodes, each leaf node containing a positive integer representing its data size.

You need to combine all leaf nodes into a single file using the minimum total cost.

The cost of merging two leaf nodes is equal to the sum of their sizes.

Each time you merge two files, the resulting file is pushed back to the set of available files.

Your task is to find the minimum total cost to merge all the files into one. This problem is analogous to constructing an Optimal Binary Merge Tree (OBMT) — or equivalently, the Huffman coding tree — where merging with minimal total cost is achieved using a min-heap (priority queue).

Explanation

Let’s say you have leaf sizes:

[5, 3, 8, 2]

Pick the two smallest → 2 and 3 → merge cost = 2 + 3 = 5.

New list → [5, 5, 8]

Pick the two smallest → 5 and 5 → merge cost = 5 + 5 = 10.

New list → [8, 10]

Pick the two smallest → 8 and 10 → merge cost = 8 + 10 = 18.
Total cost = 5 + 10 + 18 = 33
Minimum total merge cost = 33

Input Format

N
a1 a2 a3 ... aN
N → number of leaf nodes (1 ≤ N ≤ 10^5)
ai → integer size of each leaf (1 ≤ ai ≤ 10^6)

Output Format

Minimum total cost

Sample Input1

4
5 3 8 2

Sample Output1

33

Sample Input2

3
10 20 30

Sample Output2

90

Explanation

Merge 10 & 20 → cost = 30 → new list [30, 30]
Merge 30 & 30 → cost = 60
Total = 30 + 60 = 90

Task 14: (Level: Hard)

Real-Time Stock Price Tracker & Analyzer

Scenario: A fintech startup needs to monitor live stock price streams, track top movers and send alerts.

Internal tasks:

1. Use an array or list to store incoming price updates and apply sorting (e.g., merge sort or quick sort) for top gainers/losers.
2. Use a min-heap or max-heap priority queue to keep track of top N stocks efficiently as data updates.
3. Maintain a BST for quick look-up of any given stock's history, and provide find-min/find-max operations.

Task 15: (Level: Hard)

Chatbot with Real-Time Response Analysis

Scenario: A customer-service chatbot that processes incoming messages and provides responses, tracking priority issues.

Internal tasks:

1. Use queue (FIFO) to process incoming user messages; use stack for rollback or to manage recent conversations.

2. Use hash map to map keywords → responses; implement search and update operations.

3. Use priority queue to elevate urgent queries (e.g., containing “urgent”, “error”) and ensure they are handled first.

Task 16: (Level: Hard)

Given a string A, we may represent it as a binary tree by partitioning it to two non-empty substrings recursively.

Below is one possible representation of A = “great”:

```
    great
   /    \
  gr    eat
 / \    /  \
g   r  e   at
           / \
          a   t
```

To scramble the string, we may choose any non-leaf node and swap its two children.

For example, if we choose the node “gr” and swap its two children, it produces a scrambled string “rgeat”.

```
  rgeat
   /    \
  rg    eat
 / \    /  \
r   g  e   at
           / \
          a   t
```

We say that “rgeat” is a scrambled string of “great”.

Similarly, if we continue to swap the children of nodes “eat” and “at”, it produces a scrambled string “rgtae”.

```
    rgtae
   /    \
  rg    tae
 / \    /  \
r   g  ta  e
       / \
      t   a
```

We say that “rgtae” is a scrambled string of “great”.

Given two strings A and B of the same length, determine if B is a scrambled string of S.

Input Format:

The first argument of input contains a string A.

The second argument of input contains a string B.

Output Format:

Return an integer, 0 or 1:

=> 0 : False

=> 1 : True

Constraints:

1 <= len(A), len(B) <= 50

Sample Input 1:

A = " rgtae "
B = "great"

Sample Output 1:

1

Sample Input 2:

A = "hello"
B = "hel"

Sample Output 2:

0

Task 17: (Level: Hard)

Online Voting / Polling System

Scenario: A website runs polls or elections and wants to manage votes, handle unique voters, show results.

Internal tasks:

1. Use HashSet to track unique voters (prevent duplicates); use HashMap to map candidates → vote count.
2. Use quick/merge sort to sort candidates by vote count to display top winners.
3. Use a priority queue to track upcoming or most-popular polls; manage insertion/deletion of polls dynamically.

Task 18: (Level: Hard)

Minimum Cost to Connect All Cities

You are given N cities (numbered 1 to N) and M bidirectional roads.

Each road connects two cities u and v and has a construction cost w.

Your task is to find the minimum total cost required to connect all cities — that is, every city should be reachable from every other city. If it’s not possible to connect all cities, print -1. This problem represents finding the Minimum Spanning Tree (MST) of a graph.

Explanation:

- We can use Kruskal’s Algorithm with Disjoint Set Union (DSU) to efficiently find the MST.
- Sort all edges by their cost.
- Use DSU to add edges one by one (avoid cycles).
- Keep adding edges until all cities are connected.

Input Format

N M

u1 v1 w1

u2 v2 w2

...

uM vM wM

N → number of cities (1 ≤ N ≤ 10^5)

M → number of roads (1 ≤ M ≤ 2×10^5)
ui, vi → cities connected by the i-th road
wi → cost of that road

Output Format

Minimum total cost to connect all cities. If it's not possible to connect all, print -1.

Sample Input

```
4 5
1 2 5
1 3 10
2 3 4
2 4 11
3 4 2
```

Sample Output

```
11
```

Explanation

The edges in the MST are:
(3–4) → 2
(2–3) → 4
(1–2) → 5
Total cost = 2 + 4 + 5 = 11

Task 19: (Level: Easy)

Write a method to print the characters which occur a minimum number of times in the array. In case of a dilemma, print the char which is alphabetically earlier.
Consider the input as "ABCC".
The output is 'A' as it occurs a minimum number of times.
Please note that 'A' and 'B' occur the same number of times but 'A' is earlier than 'B'.

Sample Test Cases:

Sample Input:

```
ABCC
```

Sample Output:

```
A
```

Task 20: (Level: Hard)

Social Media Friend Recommendation Engine

Scenario: A social network needs to suggest new friend connections to users.
Internal tasks:

1. Model users and friendships as a graph (adjacency matrix or adjacency list).
2. Use BFS/DFS to compute "friends-of-friends" and mutual connections.
3. Use a priority queue (heap) to rank suggestions by strength of connection (mutual friend count) and display top K.

Task 21: (Level: Hard)

Campus Navigation & Way-finding System

Scenario: A university wants an app that helps students navigate between buildings, labs, facilities.

Internal tasks:

1. Model the campus as a graph (nodes = buildings/rooms, edges = paths) using adjacency matrix/list.
2. Implement BFS/DFS to explore reachable locations from a given start point; compute shortest unweighted path.
3. Use Java Collections (List, Set, and Map) to maintain visited nodes, path history, and map node IDs to building names; compare performance.

Task 22: (Level: Hard)

Travel Planner / Route Finder

Scenario: A travel-app that finds routes between cities, suggests trip plans.

Internal tasks:

1. Represent cities and routes as a graph; store routes (edges) and cities (nodes).
2. Use BFS/DFS to find reachable cities; maybe implement shortest path in unweighted graphs.
3. Sort possible routes by number of stops or cost using sorting algorithms and present top options.

Task 23: (Level: Hard)

Contact Management & Search System

Scenario: A company wants a digital phonebook/contact system with fast search and sorting.

Internal tasks:

- Use array or linked list to store contacts; implement CRUD operations (add, delete, update, and retrieve).
- Sort contacts by name or last update using insertion/selection/quick sort; implement search via binary and linear search.
- Enhance with a hash map (HashMap) for lookup by phone number; analyze the time complexity differences between list vs. hash-map lookup.

Task 24: (Level: Hard)

Maze / Labyrinth Solver

Scenario: A game or robotics simulation navigates a maze from start to exit.

Internal tasks:

- Model the maze as a 2D-array (matrix) where walls = blocked cells, open cells = nodes.
- Use BFS/DFS to find a path from start to finish (or all possible paths).
- Use backtracking to explore and mark dead-ends, count the number of unique paths; compare stack vs. recursion implementations.

Task 25: (Level: Hard)

Expense Tracker / Budget Management

Scenario: A personal finance app where users log expenses, sort and filter them, get analytics.

Internal tasks:

1. Use array/list to store expense entries (date, category, and amount).

2. Sort expenses by date or amount using quick sort/merge sort; implement search for entries by category/date.
3. Use HashMap category → total amount; use tree structure (TreeMap) to maintain sorted categories; present top expense categories via priority queue.

Task 26:(Level: Hard)

File / Data De-duplication Tool

Scenario: A cloud storage provider wants to detect duplicate files and save space by de-duplication.

Internal tasks:

1. Use hash map (file-hash → list of files) to detect duplicates; support insert and delete.
2. Use sorting algorithms to sort files by size or last-modified date for efficient checking.
3. Use tree or priority queue to list largest duplicates removed, track space saved; implement find-max and delete operations.

Task 27:(Level: Hard)

Multiplayer Online Game Leader board & Event System

Scenario: An online game maintains player scores, sorts leader boards, and handles events and matchmaking.

Internal tasks:

. Use array/list for players and their scores; implement quick sort/merge sort for leader board.

. Use priority queue to manage event scheduling or matchmaking, where top players get priority in events.

. Use graph structure for matchmaking (players = nodes, edges = connections or past matches); use BFS/DFS to find potential match clusters or teams.

Task 28 : (Level: Easy)

An automobile company manufactures both a two wheeler (TW) and a four wheeler (FW). A company manager wants to make the production of both types of vehicle according to the given data below:

- 1st data , Total number of vehicle (two – wheeler + four - wheeler)=v
- 2nd data, Total number of wheels=w

The task is to find how many two-wheelers as well as four-wheelers need to manufacture as per the given data. Print "INVALID INPUT", if inputs did not meet the constraints.

Constraints:

- 2<=W
- W%2=0
- V<W

Sample Test Cases:

Sample Input:

200 540

Sample Output: 130 70

Task 29: (Level: Easy)

Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack.

Input: An integer n, the number of items. Two lists: weights and values. An integer W, the maximum capacity of the knapsack.

Output: Print the maximum total value. **Constraints:** 1 <= n <= 1000 1 <= W <= 10^5

Approach/Explanation: Use dynamic programming to solve the 0/1 knapsack problem.

Sample Test Case:

Sample Input:

n = 3

values = [60, 100, 120]

weights = [10, 20, 30]

W = 50

Sample Output:

220

Task 30: (Level: Hard)

Data Compression Tool using Huffman Coding

Scenario: A software firm needs to compress text files using efficient encoding.

Internal tasks:

1. Given character frequency count, build Huffman tree via greedy algorithm using priority queue (heap).
2. Traverse the Huffman tree (preorder/inorder/postorder) to assign codes to characters.
3. Implement encoding and decoding of text; compare size of encoded vs. original; analyze time/space complexity.

TOTAL: 30 Hours

H. Learning Resources

i. Text Books:

1. Data Structures and Algorithms in Java: A Project-Based Approach — Dan S. Myers, 1st Edition, Cengage Learning, 2024
2. Data Structures & Algorithms in Java (6th Ed) — Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, Wiley India, 2023 (Indian Edition).
3. Data Structures & Algorithms in Java (2nd Ed) — Oswald Campesato, Packt Publishing, 2022.
4. Data Structures & Algorithms Made Easy in Java — Narasimha Karumanchi, CareerMonk Publications, 2023.
5. M. A. Weiss, Data Structures and Algorithm Analysis in Java, 3rd ed., Pearson Education, New Delhi, India, ISBN 978-8131760635.
6. M. T. Goodrich, R. Tamassia, M. H. Goldwasser, and S. Banerjee, Data Structures and Algorithms in Java, 6th ed., Indian Adaptation, Wiley India Pvt. Ltd., 2022, ISBN 978-9354247934.

ii. Reference Books:

1. Introduction to Java Programming and Data Structures, Comprehensive Version —

Y. Daniel Liang, 12th Edition, Pearson Education, 2023.

2. Data Structures and Algorithms in Java (Comprehensive) — S. K. Srivastava & Deepali Srivastava, BPB Publications, 2026.

iii. Online Resources:

1. “Competitive Programming - A Complete Guide”, [Online] Last Updated: 22.08.2025. Available: https://www.geeksforgeeks.org/dsa/competitive-programming-a-complete-guide/
2. w3schools.com, Explore Java programming language, [Online] Last Updated: 22.08.2025. Available: w3schools.com