Kurukshetra University, Kurukshetra

(Established by the State Legislature Act-XII of 1956)
("A++" Grade, NAAC Accredited)

Syllabus
for

Post Graduate Programme

M.Sc. Computer Science (Software)

as per NEP-2020
Curriculum and Credit Framework for Postgraduate Programme

With Multiple Entry-Exit, Internship and CBCS-LOCF
With effect from the session 2024-25 (in phased manner)

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS
FACULTY OF SCIENCES

KURUKSHETRA UNIVERSITY, KURUKSHETRA -136119

596

CC-1 Mathematical Foundation of Computer Science

With effect from Session: 2024-25			
Part A - Introduction			
Name of the Programme	M. Sc. Computer Science (Software)		
Semester	1^{st}		
Name of the Course	Mathematical Foundation of Computer Science		
Course Code	M24-CSE-101		
Course Type	CC-1		
Level of the course (As per Annexure-I	400-499		
Pre-requisite for the course (if any)	-		
Course Objectives	This course aims to develop a deep understanding of mathematical logic, including propositional calculus, semantics, and formal systems. It covers combinatorial methods, including counting principles, generating functions, and proof techniques like mathematical induction. Additionally, students will explore structured sets and algebraic systems, such as groups, rings, fields, and their applications in graph theory and tree properties.		
Course Learning Outcomes (CLO) After completing this course, the learner will be able to:	CLO-1. Understand and apply the principles of propositional calculus, including syntax, semantics, truth tables, validity, satisfiability, and formal deduction systems. CLO-2 Master basic counting techniques, recurrence relations, proof techniques, and the principles of structured sets, including algebraic systems and Boolean algebra. CLO-3 Comprehend and utilize graph terminologies, types of graphs, and related concepts like Euler and Hamiltonian graphs to solve relevant problems. CLO-4 Analyze and apply properties of trees, including types, spanning trees, fundamental circuits, cut-sets, connectivity, and separability to solve related problems.		
Credits	Theory	Practical	Total
	4	0	4
Teaching Hours per week	4	0	4
Internal Assessment Marks	30	0	30
End Term Exam Marks	70	0	70
Max. Marks	100	0	100
Examination Time	3 hours		

Part B- Contents of the Course

Instructions for Paper- Setter: The examiner will set 9 questions asking two questions from each unit and one compulsory question by taking course learning outcomes (CLOs) into consideration. The compulsory question (Question No. 1) will consist at least 4 parts covering entire syllabus. The examinee will be required to attempt 5 questions, selecting one question from each unit and the compulsory question. All questions will carry equal marks.

Unit	Topics	Contact Hours
I	Mathematical Logic: Propositional calculus – propositions and connectives, syntax; Semantics – truth assignments and truth tables, validity and satisfiability, tautology; Adequate set of connectives; Equivalence and normal forms; Compactness and resolution; Formal reducibility – natural deduction system and axiom system; Soundness and completeness.	15
II	Combinatorics: Basic counting sum and product, balls and bins problems, generating functions, recurrence relations. Proof techniques, principle of mathematical induction, pigeonhole principle. Structured Sets: Set, relation – Algebraic System: Groups, Semi groups, monoid, homomorphism, cosets, Ring and Field (definition), Relation, Equivalence relations, Poset, Lattices, Hasse diagram, Boolean algebra.	15
III	Graph Theory: Introduction – Graph Terminologies – Types of Graphs – Sub Graph- Multi	15

Chairman
Deptt. of Computer Science
and Applications
K.U. Kurukshetra

597

	Graph – Regular Graph – Isomorphism –Isomorphic Graphs – Sub-graph – Euler graph – Hamiltonian Graph – Related problems.	
IV	Tr Trees –Properties- Distance and Centres – Types – Rooted Tree—Tree Enumeration Labeled Tree – Unlabeled Tree –Spanning Tree – Fundamental Circuits- Cut Sets – Properties – Fundamental Circuit and Cut-set- Connectivity-Separability – Related problems.	15
	Total Contact Hours	60

Suggested Evaluation Methods

Internal Assessment: 30		**End Term Examination: 70**	
➢ **Theory**	**30**	➢ **Theory**	**70**
• Class Participation:	5	Written Examination	
• Seminar/presentation/assignment/quiz/class test etc.:	10		
• Mid-Term Exam:	15		

Part C-Learning Resources

Reference Books:

1) Rosen, K. H. (2011). *Discrete Mathematics and its Applications* (7th ed., Special Indian Edition). Tata McGraw Hill Pub. Co. Ltd.
2) Liu, C. L. (2017). *Elements of Discrete Mathematics* (2nd ed.). McGraw Hill.
3) Grimaldi, R. P. (2007). *Discrete and Combinatorial Mathematics: An Applied Introduction* (4th ed.). Pearson Education Asia.
4) Lipschutz, S., & Lipson, M. (2010). *Discrete Mathematics* (3rd ed.). Schaum's Outlines, Tata McGraw Hill Pub. Co. Ltd.
5) Koshy, T. (2006). *Discrete Mathematics with Applications*. Elsevier Publications.

598

CC-2 Advanced Computer Architecture

With effect from Session: 2024-25			
Part A - Introduction			
Name of the Programme	M. Sc. Computer Science (Software)		
Semester	1st		
Name of the Course	Advanced Computer Architecture		
Course Code	M24-CSE-102		
Course Type	CC-2		
Level of the course (As per Annexure-I	400-499		
Pre-requisite for the course (if any)	-		
Course Objectives	This course covers computational models, parallel architectures, and their relationship with programming languages. It explores parallel processing, focusing on instruction-level parallelism (ILP), pipelined processing, and superscalar processors, including their structure, scheduling, and branch handling techniques. Additionally, it examines MIMD architectures, interconnection networks, and cache coherence protocols, emphasizing performance metrics and solutions to coherence problems in multiprocessor systems.		
Course Learning Outcomes (CLO) After completing this course, the learner will be able to:	CLO-1. learn the concepts of parallel architectures and exploitation of parallelism at instruction level; CLO-2. understand architectural features of multi-issue ILP processors; CLO-3. learn MIMD architectures and interconnection networks used in them and evaluate their comparative performances; CLO-4. analyze causes of cache coherence problem and learn algorithm for its solution.		
Credits	Theory	Practical	Total
	4	0	4
Teaching Hours per week	4	0	4
Internal Assessment Marks	30	0	30
End Term Exam Marks	70	0	70
Max. Marks	100	0	100
Examination Time	3 hours		
Part B- Contents of the Course			

Instructions for Paper- Setter: The examiner will set 9 questions asking two questions from each unit and one compulsory question by taking course learning outcomes (CLOs) into consideration. The compulsory question (Question No. 1) will consist at least 4 parts covering entire syllabus. The examinee will be required to attempt 5 questions, selecting one question from each unit and the compulsory question. All questions will carry equal marks.

Unit	Topics	Contact Hours
I	Computational Model: Computational models, computer architecture, Classification of parallel architectures, Relationships between programming languages and parallel architectures. Parallel Processing: Types and levels of parallelism, Instruction Level Parallel (ILP) processors, dependencies between instructions, principle and general structure of pipelines, performance measures of pipeline, pipelined processing instructions. Code Scheduling for ILP- Processors: Basic block scheduling, loop scheduling, global scheduling.	15
II	Superscalar Processors: Emergence of superscalar processors, Tasks of superscalar processing – parallel decoding, superscalar instruction issue, shelving, register renaming, parallel execution, preserving sequential consistency of instruction execution and exception processing, comparison of VLIW & superscalar processors. Branch Handling: Branch problem, Approaches to branch handling – delayed branching, branch detection and prediction schemes, branch penalties and schemes to reduce them, multiway branches.	15

III	MIMD Architectures: Concepts of distributed and shared memory MIMD architectures, UMA, NUMA, CC-NUMA & COMA models of multiprocessors. Direct Interconnection Networks: Metrics for performance measures; Interconnection topologies: Linear Array, Ring, Chordal Rings, Star, Tree, 2D Mesh, Barrel Shifter, Hypercube.	15
IV	Dynamic interconnection networks: single shared buses, comparison of bandwidths of locked, pended & split transaction buses, arbiter logics, crossbar, multistage networks – omega. Cache Coherence Protocols: Cache Coherence Problem, Causes, Hardware Based Protocols – Snoopy Cache Protocol, Directory Schemes – Full-Map Directory, Limited Directory, Chained Directory.	15
	Total Contact Hours	60

Suggested Evaluation Methods

Internal Assessment: 30		**End Term Examination: 70**	
➢ **Theory**	**30**	➢ **Theory**	**70**
• Class Participation:	5	Written Examination	
• Seminar/presentation/assignment/quiz/class test etc.:	10		
• Mid-Term Exam:	15		

Part C-Learning Resources

Reference Books:

1) Sima, D., Fountain, T., & Kacsuk, P. (2007). *Advanced Computer Architecture*. Pearson Education.
2) Patterson, D. A., & Hennessy, J. L. (2013). *Computer Architecture – A Quantitative Approach*. Elsevier India.
3) Hwang, K. (2001). *Advanced Computer Architecture*. McGraw Hill.
4) Carter, N. (2007). *Computer Architecture*. McGraw Hill.
5) Jordan, H. F., & Alaghband, G. (2003). *Fundamentals of Parallel Processing*. Pearson Education.

Chairman
Deptt. of Computer Science
and Applications
K.U. Kurukshetra

CC-3 Advanced Data Structures and Algorithms

With effect from Session: 2024-25

Part A - Introduction

Name of the Programme	M. Sc. Computer Science (Software)
Semester	1st
Name of the Course	Advanced Data Structures and Algorithms
Course Code	M24-CSE-103
Course Type	CC-3
Level of the course (As per Annexure-I	400-499
Pre-requisite for the course (if any)	-
Course Objectives	This course covers advanced data structures such as AVL, Splay, B-Trees, Red-Black Trees, and various heap types, alongside fundamental graph representations and algorithms like DFS, BFS, and shortest path algorithms. It delves into memory allocation techniques, dynamic programming problems, and key graph algorithms for minimum spanning trees and network flow. Additionally, the course introduces NP-completeness and approximation techniques, while also exploring parallel algorithms, including parallel sorting, graph algorithms, and their design and analysis.
Course Learning Outcomes (CLO) After completing this course, the learner will be able to:	CLO-1. Master the implementation and applications of advanced tree and heap data structures for efficient data management. CLO-2. Develop a deep understanding of advanced graph algorithms and memory allocation techniques for optimized resource management. *CLO-3.* Apply advanced graph and dynamic programming algorithms to solve complex computational problems efficiently. CLO-4. Explore the development of approximation algorithms and understand the principles of parallel computing for enhanced computational performance.

Credits	Theory	Practical	Total
	4	0	4
Teaching Hours per week	4	0	4
Internal Assessment Marks	30	0	30
End Term Exam Marks	70	0	70
Max. Marks	100	0	100
Examination Time	3 hours		

Part B- Contents of the Course

Instructions for Paper- Setter: The examiner will set 9 questions asking two questions from each unit and one compulsory question by taking course learning outcomes (CLOs) into consideration. The compulsory question (Question No. 1) will consist at least 4 parts covering entire syllabus. The examinee will be required to attempt 5 questions, selecting one question from each unit and the compulsory question. All questions will carry equal marks.

Unit	Topics	Contact Hours
I	Trees: AVL Trees, Splay Trees, B-Trees, Red-Black Trees; Heaps: Binomial Heaps, Fibonacci Heaps, Pairing Heaps	15
II	Graphs: Graph Representations, Depth-First Search (DFS), Breadth-First Search (BFS), Topological Sorting, Strongly Connected Components (SCC); Memory Allocation: Buddy System, Memory Pool Allocation, Garbage Collection Algorithms	15
III	Graph Algorithms: Shortest Path Algorithms (Dijkstra's, Bellman-Ford, Floyd-Warshall), Minimum Spanning Tree (Kruskal's, Prim's), Network Flow (Ford-Fulkerson, Edmonds-Karp); Dynamic Programming: Matrix Chain Multiplication, Longest Common Subsequence (LCS), Knapsack Problems, Traveling Salesman Problem (TSP)	15
IV	Approximation Algorithms: Introduction to NP-Completeness, Approximation Techniques (Greedy, Local Search, Linear Programming), Specific Problems (Vertex Cover, Traveling	15

Chairman
Deptt. of Computer Science
and Applications
K.U. Kurukshetra

601

Salesman Problem, Set Cover); Parallel Algorithms: Introduction to Parallel Computing, Parallel Sorting (Bitonic Sort, Parallel Merge Sort), Parallel Graph Algorithms (Parallel BFS, Parallel DFS), Parallel Algorithm Design and Analysis			
		Total Contact Hours	60
Suggested Evaluation Methods			
Internal Assessment: 30		**End Term Examination: 70**	
➢ **Theory**	**30**	6) **Theory**	**70**
• Class Participation:	5	Written Examination	
• Seminar/presentation/assignment/quiz/class test etc.:	10		
• Mid-Term Exam:	15		
Part C-Learning Resources			

Reference Books:

1) Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

2) Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1983). *Data Structures and Algorithms*. Addison-Wesley.

3) Kleinberg, J., & Tardos, É. (2006). *Algorithm Design*. Pearson.

4) Tarjan, R. E. (1983). *Data Structures and Network Algorithms*. SIAM.

5) Motwani, R., & Raghavan, P. (1995). *Randomized Algorithms*. Cambridge University Press.

6) Sedgewick, R. (2011). *Algorithms* (4th ed.). Addison-Wesley.

7) Goodrich, M. T., & Tamassia, R. (2010). *Algorithm Design and Applications*. Wiley.

CC-4 Object-Oriented Programming with Java

With effect from Session: 2024-25			
Part A - Introduction			
Name of the Programme	M. Sc. Computer Science (Software)		
Semester	1^{st}		
Name of the Course	Object-Oriented Programming with Java		
Course Code	M24-CSE-104		
Course Type	CC-4		
Level of the course (As per Annexure-I	400-499		
Pre-requisite for the course (if any)	-		
Course Objectives	This course introduces Java, covering its history, features, and applications. Students will learn Java programming basics, including syntax, control flow, methods, arrays, classes, objects, inheritance, polymorphism, interfaces, and packages. Advanced topics include exception handling, file handling, multithreading, event handling, generics, JDBC, and GUI programming with Swing, preparing students for practical Java application development.		
Course Learning Outcomes (CLO) After completing this course, the learner will be able to:	CLO-1. Understand Java's background, features, and apply fundamental programming concepts including variables, operators, control flow, methods, and arrays. CLO-2 Master object-oriented programming principles including classes, objects, inheritance, polymorphism, interfaces, and packaging in Java. CLO-3 Gain proficiency in handling exceptions, working with files, implementing multithreading, and utilizing Java Collections for efficient data management. CLO-4 Explore and utilize advanced Java features such as generics, lambda expressions, JDBC for database connectivity, and GUI programming with JavaFX or Swing.		
Credits	Theory	Practical	Total
	4	0	4
Teaching Hours per week	4	0	4
Internal Assessment Marks	30	0	30
End Term Exam Marks	70	0	70
Max. Marks	100	0	100
Examination Time	3 hours		
Part B- Contents of the Course			

Instructions for Paper- Setter: The examiner will set 9 questions asking two questions from each unit and one compulsory question by taking course learning outcomes (CLOs) into consideration. The compulsory question (Question No. 1) will consist at least 4 parts covering entire syllabus. The examinee will be required to attempt 5 questions, selecting one question from each unit and the compulsory question. All questions will carry equal marks.

Unit	Topics	Contact Hours
I	Introduction to Java: History, features, and applications; Basics of Java programming: Syntax, variables, data types, operators, expressions, and statements; Control flow: Decision-making statements (if, else-if, switch), looping statements (for, while, do-while), and branching; Methods: Declaring methods, passing parameters, method overloading, and recursion; Arrays: Declaring, initializing, and manipulating arrays. Array operations and algorithms.	15
II	Classes and Objects: Declaring classes, creating objects, constructors, and instance variables; Encapsulation: Access modifiers (public, private, protected, default), getters, and setters; Inheritance: Extending classes, method overriding, super keyword, and method overloading; Polymorphism: Method overriding, dynamic method dispatch, and abstract classes; Interfaces: Defining interfaces, implementing interfaces, and using interface references; Packages: Creating and using packages, importing classes and packages.	15

Chairman
Deptt. of Computer Science
and Applications
K.U. Kurukshetra

603

III	Exception Handling: Understanding exceptions, try-catch block, throw and throws keywords, and finally block; File Handling: Reading from and writing to files using FileInputStream, FileOutputStream, FileReader, and FileWriter; Multithreading: Creating threads, thread lifecycle, synchronization, thread communication. Applet programming, Applet life Cycle, Applet Graphics programming.	15
IV	Event Handling: AWT Classes, ActionListener, MouseListener, MouseMotionListener, Layout managers, Generics: Introduction to generics, generic classes, generic methods, Java Database Connectivity (JDBC): Connecting to databases, executing SQL queries, handling transactions, and managing resources; GUI Programming: Introduction to Swing for creating graphical user interfaces (GUIs), event handling, layout management.	15
	Total Contact Hours	60

Suggested Evaluation Methods

Internal Assessment: 30		**End Term Examination: 70**	
➢ **Theory**	**30**	➢ **Theory**	**70**
• Class Participation:	5	Written Examination	
• Seminar/presentation/assignment/quiz/class test etc.:	10		
• Mid-Term Exam:	15		

Part C-Learning Resources

Reference Books:

1) Balaguruswamy, E. (2009). *Programming with JAVA: A Primer*. Tata McGraw Hill.
2) Naughton, P., & Schildt, H. (2002). *The Complete Reference Java 2*. Tata McGraw Hill.
3) Neimeyer, P., & Peck, J. (1996). *Exploring Java*. O'Reilly.
4) Hahn, H. (1996). *Teach Yourself the Internet*. Prentice-Hall of India (P.H.I.).
5) Boone, B., & Stanek, W. (2001). *Java 2 Exam Guide*. Tata McGraw Hill.

604

PC-1 PRACTICAL-1

With effect from Session: 2024-25			
Part A - Introduction			
Name of the Programme	M. Sc. Computer Science (Software)		
Semester	1st		
Name of the Course	Practical-1		
Course Code	M24-CSE-105		
Course Type	PC-1		
Level of the course	400-499		
Pre-requisite for the course (if any)			
Course objectives	This course aims the students to learn the practical implementations of the advanced data structures and algorithms based on the paper M24-CSE-103.		
Course Learning Outcomes (CLO) After completing this course, the learner will be able to:	CLO 1: Gain proficiency in implementing and manipulating various advanced tree structures, including AVL, Splay, B-trees, and Red-Black trees. CLO 2: Develop skills in implementing and performing operations on different heap structures such as binomial, Fibonacci, and pairing heaps. CLO 3: Acquire the ability to implement and apply fundamental graph algorithms for tasks like traversal, shortest path finding, and cycle detection. CLO 4: Learn and compare various memory allocation strategies and garbage collection algorithms to effectively manage memory in software applications.		
Credits	Theory	Practical	Total
	0	4	4
Teaching Hours per week	0	8	8
Internal Assessment Marks	0	30	30
End Term Exam Marks	0	70	70
Max. Marks	0	100	100
Examination Time	0	4 hours	

Part B- Contents of the Course	
Practicals	**Contact Hours**
Practical course will consist of two components Part-A and Part-B. The examiner will set 5 questions at the time of practical examination asking 2 questions from the Part-A and 3 questions from the Part-B by taking course learning outcomes (CLO) into consideration. The examinee will be required to solve one problem from the Part-A and to write and execute 1 programs from the Part-B.	120
Part-A Problems based on the theory courses M24-CSE-103 will be solved in this part and their record will be maintained in the Practical Note Book. Direct results and questions will not be asked in this section rather exercises or applied problems based on the theory parts will be done, as identified or given by the teacher concerned.	30
Part-B 1. Implement an AVL tree. Insert the following sequence of numbers: 20, 4, 15, 70, 50, 100. Show the tree structure after each insertion and perform necessary rotations. 2. Develop a splay tree and insert the following sequence of numbers: 5, 9, 3, 1, 7, 6. Show the tree structure after each insertion and splay operation. 3. Create a B-tree of order 3 and insert the following sequence of numbers: 10, 20, 5, 6, 12, 30, 7, 17. Show the tree structure after each insertion. 4. Construct a Red-Black tree and insert the following sequence of numbers: 10, 18, 7, 15, 16, 30, 25, 40, 60, 2, 1, 70. Show the tree structure after each insertion and color adjustment. 5. Compare the performance of AVL trees and Red-Black trees by inserting 1,000,000 random integers and measuring the time taken for insertion and search	90 (Lab hours include instructions for writing programs and demonstration by a teacher and for running the programs on computer by students.)

Chairman
Deptt. of Computer Science
and Applications
K.U. Kurukshetra

878

605

operations.

6. Implement a binomial heap and perform the following operations: insert(10), insert(20), insert(5), extract-min, insert(7), extract-min. Show the heap structure after each operation.
7. Develop a Fibonacci heap and perform the following operations: insert(10), insert(20), insert(5), decrease-key(20, 2), extract-min. Show the heap structure after each operation.
8. Create a pairing heap and perform the following operations: insert(10), insert(20), insert(5), insert(15), decrease-key(20, 8), extract-min. Show the heap structure after each operation.
9. Compare the performance of binomial, Fibonacci, and pairing heaps by performing 1,000,000 insertions and 500,000 extract-min operations, measuring the time taken.
10. Use a Fibonacci heap to implement Dijkstra's algorithm for a graph with 1,000 vertices and 5,000 edges, and measure the time taken to find the shortest path.
11. Implement graph representations using both adjacency matrix and adjacency list. Create a graph with 100 vertices and 500 edges, and compare the memory usage and time taken for traversals.
12. Develop a Depth-First Search (DFS) algorithm and use it to detect cycles in a directed graph with 10 vertices and 15 edges.
13. Implement a Breadth-First Search (BFS) algorithm and use it to find the shortest path in an unweighted graph with 20 vertices and 30 edges.
14. Create an algorithm for topological sorting and apply it to a directed acyclic graph (DAG) with 10 vertices and 12 edges.
15. Implement Tarjan's algorithm to find strongly connected components (SCCs) in a directed graph with 10 vertices and 15 edges.
16. Implement the buddy system for memory allocation. Simulate the allocation and deallocation of memory blocks of sizes 64KB, 128KB, 32KB, and 256KB, and show the memory structure after each operation.
17. Develop a memory pool allocation system for fixed-size memory blocks of 64 bytes. Simulate the allocation and deallocation of 1,000 blocks and measure the time taken for these operations.
18. Compare different garbage collection algorithms (mark-and-sweep, reference counting, generational) by simulating a program that creates and destroys objects, and measure the memory usage and time taken.
19. Implement a basic mark-and-sweep garbage collector in a simulated environment with 100 objects and show the effect on memory usage after garbage collection.
20. Analyze the trade-offs between different memory allocation strategies (e.g., buddy system vs. memory pool) by simulating memory usage patterns and measuring performance.
21. Implement Dijkstra's algorithm to find the shortest path in a weighted graph with 10 vertices and 20 edges, and analyze its time complexity.
22. Develop Bellman-Ford algorithm to handle graphs with negative weight edges and demonstrate its use in detecting negative cycles in a graph with 10 vertices and 15 edges.
23. Create Floyd-Warshall algorithm for finding all pairs shortest paths in a graph with 5 vertices and 10 edges, and compare its performance with Dijkstra's algorithm.
24. Implement Kruskal's algorithm to find the minimum spanning tree of a graph with 10 vertices and 15 edges, and analyze its efficiency.
25. Develop Prim's algorithm for minimum spanning trees and compare its performance with Kruskal's algorithm on a graph with 10 vertices and 15 edges.

Suggested Evaluation Methods

Internal Assessment: 30		**End Term Examination: 70**	
➢ **Practicum**	**30**	➢ **Practicum**	**70**
• Class Participation:	5	Lab record, Viva-Voce, write-up and execution of the programs	
• Seminar/Demonstration/Viva-voce/Lab records etc.:	10		
• Mid-Term Examination:	15		

Part C-Learning Resources

579 606

Recommended Books:

1) Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.
2) Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1983). *Data Structures and Algorithms*. Addison-Wesley.
3) Kleinberg, J., & Tardos, É. (2006). *Algorithm Design*. Pearson.
4) Tarjan, R. E. (1983). *Data Structures and Network Algorithms*. SIAM.
5) Motwani, R., & Raghavan, P. (1995). *Randomized Algorithms*. Cambridge University Press.
6) Sedgewick, R. (2011). *Algorithms* (4th ed.). Addison-Wesley.
7) Goodrich, M. T., & Tamassia, R. (2010). *Algorithm Design and Applications*. Wiley.

PC-2 PRACTICAL-2

With effect from Session: 2024-25	
Part A - Introduction	
Name of the Programme	M. Sc. Computer Science (Software)
Semester	1^{st}
Name of the Course	Practical-2
Course Code	M24-CSE-106
Course Type	PC-2
Level of the course	400-499
Pre-requisite for the course (if any)	
Course objectives	This course aims the students to learn the practical implementations of the concepts of Object Oriented programming in Java Language based on the paper M24-CSE-104.
Course Learning Outcomes (CLO) After completing this course, the learner will be able to:	CLO 1: Students will achieve a thorough grasp of Java programming language basics, including syntax, data types, control flow, and basic algorithms. CLO 2: Proficiency in Object-Oriented Programming (OOP): Learners will master essential OOP concepts such as classes, objects, inheritance, polymorphism, and encapsulation, applying them effectively in Java programming. CLO 3: Application of Advanced Java Concepts: Participants, will be able to implement advanced Java techniques including exception handling, multithreading, file I/O, and GUI programming using Swing. CLO 4: Development of Problem-Solving Skills: The course will enhance students' ability to analyze real-world problems, design Java-based solutions, and implement them efficiently, fostering strong problem-solving skills in programming contexts.

Credits	Theory	Practical	Total
	0	4	4
Teaching Hours per week	0	8	8
Internal Assessment Marks	0	30	30
End Term Exam Marks	0	70	70
Max. Marks	0	100	100
Examination Time	0	4 hours	

Part B- Contents of the Course	
Practicals	**Contact Hours**
Practical course will consist of two components Part-A and Part-B. The examiner will set 5 questions at the time of practical examination asking 2 questions from the Part-A and 3 questions from the Part-B by taking course learning outcomes (CLO) into consideration. The examinee will be required to solve one problem from the Part-A and to write and execute 2 programs from the Part-B.	120
Part-A Problems based on the theory courses M24-CSE-104 will be solved in this part and their record will be maintained in the Practical Note Book. Direct results and questions will not be asked in this section rather exercises or applied problems based on the theory parts will be done, as identified or given by the teacher concerned.	30
Part-B 1. Write a Java program that converts temperatures between Celsius and Fahrenheit based on user input using methods for conversion and input validation. 2. Implement a Java program to perform matrix addition, multiplication, and transpose operations using arrays and methods. 3. Develop a Java program that converts a decimal number to its binary, octal, and hexadecimal equivalents using loops and methods. 4. Create a Java program to simulate a simple bank account management system with features like deposit, withdrawal, and balance inquiry using classes, objects, and encapsulation. 5. Write a Java program that reads a text file, counts the occurrences of each word,	90 (Lab hours include instructions for writing programs and demonstration by a teacher and for running the programs on computer by students.)

808

and displays the top N most frequent words using HashMap for storage and sorting.

6. Implement a Java program to generate the first N prime numbers using a combination of loops, methods, and optimizations like the Sieve of Eratosthenes algorithm.
7. Develop a Java program that takes a month and year as input and prints the calendar for that month using control flow statements and loops for date calculation.
8. Write a Java program that generates different number patterns like pyramid patterns using nested loops and methods for pattern printing.
9. Create a Java program to manage an employee payroll system with features for adding employees, calculating salaries based on hours worked or monthly salary, and generating pay slips using classes, inheritance, and polymorphism.
10. Implement Java programs to compare the performance of different sorting algorithms (like quicksort, mergesort, and heapsort) on large arrays of integers, measuring and analyzing time complexity.
11. Develop a Java program that recursively searches a directory for files matching a given pattern and displays the file paths using recursion and file handling classes.
12. Write a Java program to perform arithmetic operations (addition, subtraction, multiplication, division) on large numbers using BigInteger class and exception handling for division by zero.
13. Implement a Java program to solve the Tower of Hanoi problem for N disks using recursion, demonstrating the steps and movements required.
14. Write a Java program to find the largest and smallest elements in an array.
15. Implement a Java program to sort an array of integers using bubble sort.
16. Create a Java program to find the frequency of each element in an array.
17. Develop a Java program to reverse an array without using an additional array.
18. Write a Java program to merge two sorted arrays into a single sorted array.
19. Define a Java class representing a Student with private instance variables and public getter and setter methods.
20. Create a Java program to demonstrate constructor overloading in a class.
21. Implement a Java program to calculate the area and perimeter of a rectangle using a class and object.
22. Develop a Java program to implement inheritance by creating a base class Animal and derived classes like Dog and Cat.
23. Write a Java program to demonstrate method overriding by implementing a base class Shape and derived classes like Circle and Rectangle.

Suggested Evaluation Methods

Internal Assessment: 30		**End Term Examination: 70**	
➢ **Practicum**	**30**	➢ **Practicum**	**70**
• Class Participation:	5	Lab record, Viva-Voce, write-up and execution of the programs	
• Seminar/Demonstration/Viva-voce/Lab records etc.:	10		
• Mid-Term Examination:	15		

Part C-Learning Resources

Recommended Books:

1) Balaguruswamy, E. (2009). *Programming with JAVA: A Primer*. Tata McGraw Hill.
2) Naughton, P., & Schildt, H. (2002). *The Complete Reference Java 2*. Tata McGraw Hill.
3) Neimeyer, P., & Peck, J. (1996). *Exploring Java*. O'Reilly.
4) Hahn, H. (1996). *Teach Yourself the Internet*. Prentice-Hall of India (P.H.I.).
5) Boone, B., & Stanek, W. (2001). *Java 2 Exam Guide*. Tata McGraw Hill.

Chairman
Deptt. of Computer Science
and Applications
K.U. Kurukshetra

509