B.Tech. Programme in
COMPUTER SCIENCE AND ENGINEERING

(IV semester course structure and syllabi)

COLLEGE OF ENGINEERING
(AUTONOMOUS)

2019 Regulations

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING

(AUTONOMOUS)

AFFILIATED TO JNTU- KAKINADA

MADHURAWADA, VISAKHAPATNAM

R - 19

Course Structure for IV Semester

COMPUTER SCIENCE AND ENGINEERING					
Course Code	**Name of the Course**	**L**	**T**	**P**	**C**
19BM1110	Numerical Methods	3	1	0	4
19CT1107	Formal Languages & Automata Theory	3	0	0	3
19CT1108	Object Oriented Programming through JAVA	3	0	0	3
19CS1103	Algorithms and Analysis	3	0	2	4
19CS1104	Principles of Operating Systems	3	0	2	4
19ME1110	Design Thinking and Innovation	1	0	2	2
19CT1109	Object Oriented Programming through JAVA Lab	0	0	3	1.5
	TOTAL	**16**	**1**	**9**	**21.5**

NUMERICAL METHODS
(CSE)

Course Code: 19BM1110

L	T	P	C
3	1	0	4

Course Objectives:

The objective of this course is to familiarize the students with numerical methods of solving the nonlinear equations, interpolation, differentiation, integration, and ordinary differential equations.

Course Outcomes:

At the end of the course, the student will be able to:

CO1: calculate a root of algebraic and transcendental equations. Explain relation between the finite difference operators.

CO2: compute interpolating polynomial for the given data.

CO3: solve ordinary differential equations numerically using Euler's, Modified Eulermethod, RK methods.

CO4: numerically solve linear system of equations by direct and iterative methods

CO5: compute eigenvalues and eigenvectors of a matrix using different methods.

Unit I: Solutions of algebraic and transcendental equations and Finite Differences 10 Lectures

Solutions of algebraic and transcendental equations-Introduction, Bisection Method, Regula Falsi Method, Newton Raphson Method (including error analysis). Finite differences, differences of polynomial.

(Sections 2.1, 2.2, 2.3, 2.5, 3.3 and 3.5 of the textbook)

Learning Outcomes:

At the end of this unit, the student will be able to

1. determine approximate roots of an equation by using different numerical methods (L3)
2. evaluate the missing terms in given tabular data (L5)
3. explain various discrete operators and find the relation among operators (L2)

Unit II: Interpolation and Numerical Differentiation 10 Lectures

Interpolation: Introduction, Newton's forward difference interpolation, Newton's backward difference interpolation, Lagrange's interpolation and Newton's divided interpolation, Inverse interpolation.

Numerical Differentiation: Introduction, Derivatives using forward difference formula, backward difference formula.

(Sections 3.1, 3.6, 3.9.1, 3.10.1, 3.11, 6.2 [excluding 6.2.1 - 6.2.3] of the textbook)

Learning Outcomes:

At the end of this unit, the student will be able

1. apply Newton forward and backward formulas for equal and unequal intervals (L3)
2. describe how to determine values of derivatives of unknown function at the intermediate points with the given data (L2)
3. evaluate an interpolating polynomial for the given tabular data (L5)

UNIT III: Numerical Integration and Numerical solution of Ordinary Differential Equations

10 Lectures

Numerical Integration: Introduction, Trapezoidal rule, Simpson's $1/3^{rd}$ rule, Simpson's $3/8^{th}$ rules.

Numerical solution of ordinary differential equations: Introduction, Euler's method, Modified Euler method, Runge-Kutta method

(Sections 6.4.1, 6.4.2, 6.4.3, 8.1, 8.4, 8.5 of the textbook)

Learning Outcomes:

At the end of this unit, the student will be able to

1. evaluate the area bounded by non negative functions by using different numerical Methods (L5)
2. determine the solution of an ODE by Euler's method (L3)
3. illustrate the solution of an ODE using R-K method (L4)

UNIT IV: Numerical Linear Algebra -I **10 Lectures**

Solution of linear Systems (Iterative Methods) - Gauss elimination, Jacobi Iterative method; Gauss Seidel Iterative Method, LU Factorization- Doolittle's Method, Crout's Method; Cholesky's Method.

(Sections 7.5.1, 7.5.6, 7.6(a),(b) of textbook 1)

Learning Outcomes:

At the end of this unit, the student will be able to

1. solve linear simultaneous equations, by using LU Factorization (L3)
2. illustrate the solution of the linear simultaneous equations using iterative methods (L4)
3. explain Doolittle's, Crout's and Cholesky's Methods (L2)

UNIT V: Numerical Linear Algebra -II **10 Lectures**

Rayleigh's power method, Gram Schmidt process, QR Factorization of matrices, Singular Value Decomposition, Principal component analysis.

(Sections 5.8, 6.4, 7.4, 7.5 of textbook 2)

Learning Outcomes:

At the end of this unit, the student will be able to

1. determine orthonormal set of vectors from the given linear independent set of vectors(L3)
2. evaluate largest eigen value and eigen vector using Rayleigh's power method (L5)
3. determine singular values and singular vectors (L3)

Textbooks:

1. S.S. Sastry, *"Introductory Methods of Numerical Analysis",* 5th edition, Prentice Hall India Pvt. Limited, 2012.
2. David C. Lay, "*Linear Algebra and Its Applications*", 4th edition, Addison-Wesley, 2012

Reference:

1. M.K. Jain, S.R.K. Iyengar and R.K.Jain, *"Numerical Methods for Scientific and Engineering Computation",* 4th edition, New Age International (P) Limited, Publishers, 2004.
2. Samuel Daniel Conte, Carl W. De Boor, *"Elementary Numerical Analysis: An Algorithmic Approach",* 3rd edition, McGraw- Hill, 2008.
3. Gilbert Strang, "*Introduction to Linear Algebra*", 5th edition, Wellesely- Cambridge Press, 2016.

FORMAL LANGUAGES & AUTOMATA THEORY

(Common to CSE & IT)

Course Code:19CT1107	L	T	P	C
	3	0	0	3

Prerequisites: NIL

Course Outcomes:

At the end of the Course the student shall be able to
CO1: Employ finite state machines for modeling and their power to recognize the languages.
CO2: Understand the concept of Regular languages and Converting Regular Expression to Finite Automata (Vice Versa)
CO3: Understand the concept of context free languages and normal forms
CO4: Design CFG's and PDA as well for the given set of grammars
CO5: Designing turing machines for the given set of grammars.

UNIT-I **12 Lectures**

Fundamentals: Formal Languages, Strings, Alphabets, Languages, Chomsky Hierarchy of languages.

Finite Automata: Introduction to Finite State machine, Acceptance of strings and languages, Deterministic finite automaton (DFA) and Non-deterministic finite automaton (NFA), Equivalence of NFA and DFA – Equivalence of NDFAs with and without €-moves, Minimization of finite automata, Equivalence between two DFA's, Finite automata with output – Moore and Mealy machines, conversion of Moore to Mealy and Mealy to Moore

Learning Outcomes: At the end of the module the student will be able to
1. describe about Finite Automata Model (L2)
2. distinguish NFA and DFA (L4)
3. translate NFA with €-moves to NFA and DFA (L2)
4. distinguish Mealy and Moore Machines (L4)
5. translate Mealy to Moore and Vice-versa (L2)

UNIT-II **10 Lectures**

Regular Languages: Regular sets, Regular expressions, Operations and applications of regular expressions, Identity rules, Conversion of a given regular expression into a finite automaton, Conversion of finite automata into a regular expression, Pumping lemma for regular sets, Closure properties of regular sets (proofs not required).

Learning Outcomes: At the end of the module the student will be able to
1. describe about Regular Sets and Regular Expressions (L2)
2. relate Regular Expressions and Finite Automata (L4)
3. recognize Closure Properties of Regular sets (L2)
4. judge whether a language is Regular language or not (L5)

UNIT-III **(10 Lectures)**

Regular Grammars: Definition of a grammar, Regular grammars, Right linear and left linear grammars, Conversion from left linear to right linear grammars, Equivalence of regular grammar and finite automata, Inter conversion.

Context Free Grammars: Context free grammars and languages, Derivation trees, Leftmost and rightmost derivation of strings and Sentential forms, Ambiguity, left recursion and left factoring in context free grammars, Minimization of context free grammars, Normal forms for context free grammars, Chomsky normal form, Greibach normal form, Pumping Lemma for Context free Languages, Closure and decision properties of context free languages

Learning Outcomes: At the end of the module the student will be able to
1. describe about Regular Grammar and Context Free Grammar (L2)
2. distinguish Regular grammar and Context Free Grammar (L4)
3. recognize Ambiguity of Context Free Grammar (L2)
4. construct Normal Form for a Context Free Grammar (L6)
5. understand Closure Properties of Context Free Grammar (L2)
6. judge whether a language is Context Free language or not (L5)

UNIT-IV **(10 Lectures)**

Pushdown Automata: Introduction to Pushdown automata, Acceptance of context free languages, Acceptance by final state and acceptance by empty state and its equivalence, Equivalence of context free grammars and pushdown automata, Inter-conversion (Proofs not required), Introduction to deterministic pushdown automata.
Learning Outcomes: At the end of the module the student will be able to
1. design a PushDown Automata for a Language (L6)
2. translate PDA and Context Free Grammar (L2)
3. understand deterministic pushdown automata (L4)

UNIT-V **(10 Lectures)**

Turing Machine: Introduction to Turing Machine, Design of Turing machines, Types of Turing machines, Computable functions, Recursive and recursively enumerable languages, closure properties of recursive and recursively enumerable languages
Learning Outcomes: At the end of the module the student will be able to
1. design a Turing Machine for a Language (L6)
2. describe Different Types of Turing machines (L2)
3. understand the closure properties of Recursive and Recursively Enumerable languages (L2)

Text Book:

1. John E Hopcroft, Rajeev Motwani, Jeffrey D.Ullman, "*Introduction to Automata Theory Languages and Computation*", 3rd Edition, Pearson Education, 2011.

Reference Books:

1. Peter Linz, " *An introduction to Formal Languages and Automata*", 6th Edition, Jones & Bartlett, 2016
2. Mishra and Chandrashekaran, "*Theory of Computer Science – Automata Languages and Computation*", 3rd Edition, PHI, 2009
3. K.V.N.Sunitha , N.Kalyani, "*Formal Languages and Automata Theory*", 1st Edition, TMH, 2010
4. Michel Sipser, "*Introduction to Theory of Computation*", 2nd Edition, Thomson, 2012

Web Reference:

https://swayam.gov.in/nd1_noc19_cs79/preview

OBJECT ORIENTED PROGRAMMING THROUGH JAVA

(Common to CSE & IT)

Course Code: 19CT1108

L	T	P	C
3	0	0	3

Prerequisites: Problem Solving using C, Data Structures and Algorithms

Course Outcomes:

At the end of the Course the student shall be able to

CO1: Interpret object orientation and Utilize programming strategies and Contrast classes and objects

CO2: Analyze Inheritance and Dynamic Method Dispatch

CO3: Design Packages and Interpret various classes in different packages

CO4. Manage Exceptions and Apply Threads

CO5:Create GUI screens along with event handling and write network programs

UNIT-I: INTRODUCTION TO OBJECTS & CLASSES **12 Lectures**

What is Object Oriented Programming? Object Orientation as a New Paradigm: The Big Picture (TEXT BOOK-2), An Overview of Java: Process Oriented Vs Object Oriented Programming, OOP Principles, Java BuzzWords, The Byte Code, A First Simple Program.

Class Fundamentals with Variables and Methods, Declaring objects for accessing variables and methods.

Data Types and Variables, Operators and Expressions, Control Statements, Type Conversion and casting, Lexical Issues in Java, Arrays: Single Dimension, command line arguments, Arrays: Multi Dimension.

Constructors: Default and Parameterized, this keyword and Garbage Collection, Final and Static Keywords, Overloading Methods, Overloading Constructors, Using objects as Parameters, Returning objects, String and String Buffer

Learning Outcomes:

At the end of the module, students will be able to:

1. summarize object oriented programming features and Java features. (L2)
2. explain classes, objects. (L3)
3. use various data types and control statements in Java. (L2)
4. explain about constructors and methods. (L3)
5. write Java programs for manipulating Strings. (L6)

UNIT-II: INHERITANCE **08 Lectures**

Inheritance Basics, Types of Inheritance, Using Super keyword for constructors, Super to call variables and methods, Method Overriding, Dynamic Method Dispatch.

Learning Outcomes:
At the end of the module, students will be able to:
1. classify different kinds of inheritance. (L4)
2. summarize the usage of super keyword in inheritance. (L2)
3. explain Method overriding and dynamic method dispatch (L3)

UNIT-III:
PACKAGES AND INTERFACES **11 Lectures**
Defining a Package, importing a package, Package Example, Access Protection, An Access Example, Abstract classes, Interfaces: Defining and Implementing Interfaces
Exploring java.lang
Wrapper classes, Object, Math, Runtime
Exploring java.util:
The collection framework: ArrayList, HashSet and HashMap, StringTokenizer, Calendar, Random, Scanner
Exploring java.io:
File class, Byte Streams, Character Streams, FileInputStream, FileOutputStream, FileReader and FileWriter classes

Learning Outcomes:
At the end of the module, students will be able to:
1. Demonstrate the use of packages in Java. (L2)
2. Identify various built-in Java classes. (L2)
3. Apply various methods of Java built-in classes. (L3)

UNIT-IV: **10 Lectures**
Exception Handling:
Exception Handling Fundamentals, Exception Types, throw, throws and finally, Creating your own exceptions, Chained Exceptions
Multithreaded Programming:
Java Thread Model, The Main thread, Two ways of Creating a Thread, Creating Multiple Threads, isAlive(),join(), Synchronization, Inter Thread Communication

Learning Outcomes:
At the end of the module, students will be able to:
1. Distinguish between error and exception. (L4)
2. Illustrate exception handling in Java. (L3)
3. Write Java multi-threaded programs. (L6)

UNIT-V: **9 Lectures**
Introducing GUI Programming with Swings
Swing Features, MVC Connection, Components and Containers, Panes, Simple Swing Application, Simple Swing Applet, Layout Managers: Flow, Border, Card, Grid, Grid Bag, Working with Color, Working with Fonts, Painting in Swing, Exploring Swing Components.
Delegation Event Model: Event Classes, Sources and Listeners.
Exploring java.net: Socket, ServerSocket, InetAddress, DataGramSocket, URL, Client-Server Program using Sockets

Learning Outcomes:
At the end of the module, students will be able to:

1. Use swings to build GUI based applications (L3)
2. Design event driven based GUI applications. (L6)
3. Write network based Java applications (L6).

Text books:

1. Herbert Schildt, "*Java The Complete Reference*", 11th Edition, McGrawHill, 2019

2. Timothy budd, "*An Introduction to Object-oriented Programming"*, 3rd Edition, Pearson Education, 2009.

Reference book:

1. Y. Daniel Liang "*Introduction to Java Programming Comprehensive Version*" 10th Edition, Pearson, 2015

ALGORITHMS AND ANALYSIS

(Common to CSE & IT)

Course Code: 19CT1109

L	T	P	C
3	0	0	3

Course Outcomes: At the end of the Course the student shall be able to
CO1: Analyse the asymptotic performance of algorithms.
CO2: Write rigorous correctness proofs for algorithms.
CO3: Demonstrate a familiarity with major algorithms.
CO4: Apply important algorithmic design paradigms and methods of analysis.
CO5: Synthesize efficient algorithms in common engineering design situations

UNIT-I **12 Lectures**

Introduction: Algorithm, Pseudocode for expressing algorithms, Performance Analysis-Space complexity, Time complexity, Asymptotic Notation Big Oh notation, Omega notation, Theta notation and Little Oh notation, Probabilistic analysis, Amortized analysis. Disjoint Sets-disjoint set operations, union and find algorithms, spanning trees, connected components and biconnected components.

Learning Outcomes: At the end of the module, students will be able to:
1. construct asymptotic runtime bounds for reasonably straightforward pseudo-code with nested loop. (L2)
2. analyse worst-case running times of algorithms using asymptotic analysis.(L4)
3. discuss the correctness of algorithms using inductive proofs and invariants.(L2)

UNIT-II **10 Lectures**

Divide And Conquer: General method, applications-Binary search, Quick sort, Merge sort, Strassen's matrix multiplication.

Greedy Method: General method, applications-Job sequencing with deadlines, knapsack problem, Minimum cost spanning trees, Single source shortest path problem.

Learning Outcomes: At the end of the module, students will be able to:
1. describe the divide-and-conquer paradigm. (L2)
2. assess greedy algorithms, and analyze them. (L5)
3. apply the appropriate sorting techniques when an algorithmic design situation calls for it. (L4)

UNIT-III **11 Lectures**

Dynamic Programming: General method, applications-Matrix chain multiplication, Optimal binary search trees, 0/1 knapsack problem, All pairs shortest path problem, Travelling salesperson problem, Reliability design.

Learning Outcomes: At the end of the module, students will be able to:

1. describe the different methods of dynamic programming. (L2)
2. apply the knapsack problem to the given problem and their analyses. (L3)
3. construct the reliability algorithm and analyze them. (L6)

UNIT-IV **11 Lectures**

Backtracking: General method, applications-n-queen problem, sum of subsets problem, graph coloring, Hamiltonian cycles.

Branch and Bound: General method, applications- Travelling salesperson problem, 0/1 knapsack problem- LC Branch and Bound solution, FIFO Branch and Bound solution.

Learning Outcomes: At the end of the module, students will be able to:

1. analyze the Travelling salesman problem for the given problem. (L4)
2. explain the major graph algorithms and their analyses. (L2)
3. evaluate graph computations as key components, and analyze them. (L5)

UNIT-V **8 Lectures**

NP-hard And NP-complete Problems: Basic concepts, non deterministic algorithms, NP - Hard and NP Complete classes, Cook's theorem.

Learning Outcomes: At the end of the module, students will be able to:

1. apply the NP-HARD algorithm for real time applications. (L3)
2. evaluate the non deterministic algorithms and analyze them.(L5)
3. write cook's algorithm for the given problem. (L6)

Textbooks:

1. Ellis Horowitz, Sartaj Sahni and Rajasekharam, *"Fundamentals of Computer Algorithms"*,2nd Edition, University Press, 2008.

2. M.T.Goodrich and R.Tomassia, " *Algorithm Design Foundations, Analysis and Internet examples"*, 1st Edition, John wiley and sons, 2006.

Reference Books:

1. T.H.Cormen,C.E.Leiserson,R.L.Rivest,andC.Stein *"Introduction to Algorithms*", 3rd Edition, PHI / Pearson Education, 2009.

2. R.C.T.Lee, S.S.Tseng, R.C.Chang and T.Tsai, *"Introduction to Design and Analysis of Algorithms A strategic approach"*, 2nd Edition , Tata McGraw Hill,2009.

3. Allen Weiss, *"Data structures and Algorithm Analysis in C++"*, 2nd Edition, Pearson education, 2009.

4. Aho, Ullman and Hopcroft, *"Design and Analysis of Algorithms"*, 3rd Edition, Pearson education,2008.

List of Programs:

1. Applying divide and conquer method to sort n elements using quick sort.

a) Elements can be read from a file or can be generated using random number generator.

b) Determine time required to sort the elements.

c) Repeat experiment for different values of n and plot the graph of the time taken versus n.

2. Applying divide and conquer method to sort n elements using merge sort.

a) Elements can be read from a file or can be generated using random number generator.

b) Determine time required to sort the elements.

c) Repeat experiment for different values of n and plot the graph of the time taken versus n.

3. Applying divide and conquer method to sort n elements using heap sort.

a) Elements can be read from a file or can be generated using random number generator.

b) Determine time required to sort the elements.

c) Repeat experiment for different values of n and plot the graph of the time taken versus n.

4. Applying dynamic programming methodology to implement 0/1 Knapsack problem.

a) Write the function to determine optimal subset that fits into knapsack.

b) Construct the profit table.

5. Applying dynamic programming methodology to calculate the transitive closure of a directed graph.

a) Implement the aforementioned problem using Warshall's algorithm.

b) Determine the complexity of Warshall's algorithm.

6. Applying dynamic programming methodology to find all pairs shortest path of a directed graph.

a) Implement the aforementioned problem using Floyd's algorithm.

b) Determine the complexity of Floyd's algorithm.

7. Applying greedy methodology to determine minimum spanning tree.

a) Implement Prim's algorithm for an undirected graph.

b) Determine the complexity of Prim's algorithm.

8. Applying greedy methodology to determine minimum spanning tree.

a) Implement Kruskal's algorithm for an undirected graph.

b) Determine the complexity of Kruskal's algorithm.

9. Implement N Queens problem using backtracking.

a) Write a function to check whether a kth queen can be placed in a specific column or not.

b) Write a function to graphically display solution to n queens problem.

10. Applying approximation algorithm to find optimal solution for a given problem.

a) Implement the aforementioned algorithm for travelling salesman problem.

b) Determine the error in the approximation.

References:

1. Ellis Horowitz, Sartaj Sahni and Rajasekharam, *"Fundamentals of Computer Algorithms"*,2nd Edition, University Press, 2008.

Web references:

1. http://cse.iitkgp.ac.in/~abhij/course/theory/Algo1/Autumn11/
2. http://web.stanford.edu/class/cs161/

PRINCIPLES OF OPERATING SYSTEMS

Course Code: 19CS1104	L	T	P	C
	3	0	2	4

Prerequisites: Computer Organization.

Course Outcomes:

At the end of the Course the student shall be able to

CO1: Understand functional architecture of an operating system.
CO2: Compare CPU scheduling algorithms.
CO3: Analyze process coordination.
CO4: Differentiate File System and directory implementations.
CO5: Analyze different program and network threats to the system.

UNIT-I **10 Lectures**

Introduction: What is operating system, functions of operating system, types of operating system, computer system organization, computer system architecture, operating system structure, operating system operations, computing environments, open source operating systems.

Operating System Structures: operating system services, system calls, types of system calls, system programs, operating system structure, operating system debugging and system boot..

At the end of the module, students will be able to:
1. summarize operating system services, organization and architecture. (L2)
2. understand the concept of system calls. (L2)
3. analyse various computing systems(L4)

UNIT-II **10 Lectures**

Process Concept: Process, Process Control Blocks, Operations on Processes, Inter process Communication.
Multithreaded Programming: Threads, Multicore Programming, Multithreading Models. **Process Scheduling:** Scheduling Criteria, scheduling algorithms (FCFS, SJ F, Round Robin, and Priority) and their evaluation, Multiprocessor scheduling. Case Study: Linux.

At the end of the module, students will be able to:
1. explain Process concepts and Identify the operations on process (L2)
2. analyse Inter Process Communication(L4)
3. understand Multithreading(L2)
4. analyze, differentiate and apply scheduling algorithms(L4)

UNIT-III **10 Lectures**

Synchronization: The Critical- section problem, Peterson's Solution, Synchronization Hardware, semaphores, classic problems of synchronization, monitors, Synchronization examples: Windows and Linux.
Deadlocks: System model, deadlock characterization, Methods for Handling Deadlock, deadlock prevention, detection and Avoidance, recovery from deadlock.

At the end of the module, students will be able to:
1. explain the synchronization problems (L2).
2. analyse and understand the problems of Deadlocks (L4).
3. Understand the methods of handling, prevention, detection and avoidance of deadlocks (L2).

UNIT-IV **10 Lectures**

Memory management strategies: Swapping, contiguous memory allocation, segmentation, paging, structure of the page table, examples of Intel 32 and 64-bit architectures.

Virtual-memory management: Demand paging, Copy on write, page-Replacement algorithms (FIFO, LRU, LFU, Optimal Page Replacement), thrashing.

File systems and implementation: File Concept, Access Methods, Directory Structure, File System Mounting. File system structure, File System Implementation, Directory Implementation, Allocation Methods, Network file system, Free-space Management.

At the end of the module, students will be able to:
1. explain the memory management strategies (L2)
2. differentiate between paging and segmentation(L4)
3. illustrate different page replacement algorithms (L3)
3. understand the file concepts and directory structure(L2)
4. analyse virtual memory(L4)

UNIT-V **10 Lectures**

Secondary-storage structure: Overview of Mass-storage structure, disk structure, disk attachment, disk scheduling, swap-space management, RAID.

Protection And Security: Goals and Principles of Protection, Domain of protection, Access Matrix, Implementation of Access Matrix, Access control, Revocation of Access Rights, The Security problem, program threats, system and network threats

At the end of the module, students will be able to:
1. understand and apply disk scheduling concepts (L3)
2. differentiate protection and security (L4)
3. explain program, system & network threats (L2)
4. analyse security problem (L4)

Text Book:

1. Abraham Silberchatz, Peter B. Galvin, Greg Gagne,"*Operating System Concepts*", 9th Edition, John Wiley &Sons, 2019.

References:

1. D.M. Dhamdhere, "*Operating systems - A Concept based Approach"*, 2nd Edition, TMH.
2. Charles Crowley, "*Operating Systems - A Design Approach"*, 1stEdition, TMH.
3. Andrew S Tanenbaum, "*Modern Operating Systems",* 3rd Edition, Pearson/PHI.
4. William Stallings, "*Operating Systems – Internal and Design Principles"*, 6th Edition, Pearson education/PHI.

List of Programs:

1. Write a program to simulate the following non-preemptive CPU scheduling algorithms to find turnaround time and waiting time.

a) FCFS b) SJF c) Round Robin (pre-emptive) d) Priority

2. Write a program to simulate multi-level queue scheduling algorithm

3. Write a program to simulate producer-consumer problem using semaphores.

4. Write a program to simulate the concept of Dining-Philosophers problem.

5. Write a program to simulate Bankers algorithm for the purpose of deadlock avoidance.

6. Write a program to simulate the following contiguous memory allocation techniques

a)Worst-fit b) Best-fit c) First-fit

7. Write a program to simulate the MVT and MFT memory management techniques

8. Write a program to simulate paging technique of memory management.

9. Write a program to simulate page replacement algorithms:

a) FIFO b) LRU c) Optimal d)LFU

10. Write a program to simulate the following file allocation strategies:

a) Sequential b) Indexed c) Linked

11. Write a program to simulate the following file organization techniques

a) Single level directory b) Two level directory c) Hierarchical

12. Write a program to simulate the following disk scheduling algorithms:

a) FCFS b) SCAN c) C-SCAN

References:

1. Abraham Silberchatz, Peter B. Galvin, Greg Gagne,*"Operating System Principles"*, 8thEdition, John Wiley &Sons.

2. William Stallings, "*Operating Systems – Internal and Design Principles*", 6th Edition, Pearson education/PHI.

DESIGN THINKING AND INNOVATION

(An activity-based course)

Course Code: 19ME1110

L	T	P	C
1	0	2	2

Course Outcomes:

After successful completion of this activity the student will be able to:

CO1: Outline a problem, apply methods of Empathy on user groups

CO2: Describe and Define the problem specific to the user group

CO3: Apply Ideation tools to generate Ideas to solve the problem

CO4: Develop prototype

CO5: Test the ideas and demonstrate Storytelling ability to present the Ideas

Students shall form into groups and Identify a problem (preferably societal problem with engineering orientation to solve) suitable for the design thinking and go through the process week-wise. At the end of each phase,* brief *documentation shall be submitted and a final report covering all phases has to be submitted at the end of the semester.

Weeks 1-3:

Introduction to Design Thinking: A primer on design thinking - Traditional approach, The new design thinking approach. Stages in Design Thinking: Empathize, Define, Ideate, Prototype, Test. Mindset for design thinking, Design thinking for product and process innovation, Difference between engineering design and design thinking.

Case Studies: General, Engineering and Service applications.

Activities: Identify an Opportunity and Scope of the Project

Explore the possibilities and Prepare design brief

Weeks 4-6:

Methods and Tools for Empathize and Define phases:

Empathize - Methods of Empathize Phase: Ask 5 Why / 5W+H questions, Stakeholder map, Empathy Map, Peer observation, Trend analysis

Define - Methods of Define Phase: Storytelling, Critical items diagram, Define success

Activities: Apply the methods of empathize and Define Phases

Finalize the problem statement

Weeks 7-8:

Methods and Tools for Ideate phase:

Ideate - Brainstorming, 2X2 matrix, 6-3-5 method, NABC method;

Activities: Apply the methods of Ideate Phase: Generate lots of Ideas

Weeks 9-11:

Methods and Tools for Prototype Phase:

Prototype - Types of prototypes - Methods of prototyping - Focused experiments, Exploration map, Minimum Viable Product;

Activities: Apply the methods of Prototype Phase: Create prototypes for selected ideas

Weeks 12-13:

Methods and Tools for Test Phase:

Test - Methods of Testing: Feedback capture grid, A/B testing

Activities: Collect feedback; iterate and improve the ideas

Weeks 14-15:

Solution Overview - Create a Pitch - Plan for scaling up - Road map for implementation

Activities: Present your solution using Storytelling method

Week 16:

Project Submission: Fine tuning and submission of project report

Reference books:

1. Tim Brown, *Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation*, HarperCollins e-books, 2009.
2. Michael Lewrick, Patrick Link, Larry Leifer, *The Design Thinking Toolbox*, John Wiley & Sons, 2020.
3. Michael Lewrick, Patrick Link, Larry Leifer, *The Design Thinking Playbook*, John Wiley & Sons, 2018.
4. Kristin Fontichiaro, *Design Thinking*, Cherry Lake Publishing, USA, 2015.
5. Walter Brenner, Falk Uebernickel, *Design Thinking for Innovation - Research and Practice*, Springer Series, 2016.
6. Gavin Ambrose, Paul Harris, *Design Thinking*, AVA Publishing, 2010.
7. Muhammad Mashhood Alam, *Transforming an Idea into Business with Design Thinking*, First Edition, Taylor and Francis Group, 2019.
8. S. Balaram, *Thinking Design*, Sage Publications, 2011.

Web References:

1. https://designthinking.ideo.com/
2. https://thinkibility.com/2018/12/01/engineering-vs-design-thinking/
3. https://www.coursera.org/learn/design-thinking-innovation
4. https://swayam.gov.in/nd1_noc20_mg38/preview

OBJECT ORIENTED PROGRAMMING THROUGH JAVA LAB

(Common to CSE & IT)

Course Code: 19CT1109

L	T	P	C
0	0	3	1.5

Prerequisite: Problem Solving Using C, Data Structures and Algorithms

Course Outcomes: At the end of the Course the student shall be able to
CO1: Write basic Java applications and use arrays
CO2: Create classes, objects and apply Inheritance
CO3: Create Packages and build applications using default packages.
CO4: Manage Exceptions and develop multithreaded applications.
CO5: Create GUI applications which are event based and write network programs.

List of Experiments:

1. a) Implement the following programs using command line arguments
   i) Accept two strings from the user and print it on console with concatenation of "and" in the middle of the strings.
   ii) Accept 12-hour time zone and convert into its corresponding 24-hour time zone.
   Note: Accept hours, minutes and seconds separately from the user (e.g. 07 05 45 PM should be displayed as 19:05:45).
   iii) Accept a number 'n' and print the list of 'n' Fibonacci terms recursively.

   b) Perform the above programs using Scanner class.

2. a) Write a program that accepts the set of inputs from the user of various integer data types and determines the primitive data type that is capable of properly storing that input.

   b) Write a program that accepts an array of integers and print those which are both odd and prime. If no such element in that array print "Not found".

   c) Write a program to accept contents into an Integer Array and print the frequency of each number in the order of their number of occurrences.

3. a) Write a program that accepts an 'n' ordered square matrix elements into a single dimension array and print the elements of leading diagonal (top left to bottom right).

   b) Write a program that accepts an 'm x n' double dimension array, where 'm' represents financial years and 'n' represents Ids of the items sold. Each element in the array represents the number of items sold in a particular year. Identify the year and id of the item which has more demand.

c) Write a program that accepts an ‘n’ ordered square matrix and calculate the absolute difference between the sums of elements in their diagonals.

4. a) Create a class Box that uses a parameterized constructor to initialize the dimensions of a box. The dimensions of the Box are width, height, depth. The class should have a method that can return the volume of the box. Create an object of the Box class and test the functionality.

b) Create a new class called Calculator with the following methods:

- A static method called powerInt(int num1,int num2)
  This method should return num1 to the power num2.
- A static method called powerDouble(double num1,double num2).
  This method should return num1 to the power num2.
- Invoke both the methods and test the functionality. Also count the number of objects created.

5. a) Accept a String and a number ‘n’ from user. Divide the given string into substrings each of size ‘n’ and sort them lexicographically.

b) Accept an array of strings and display the number of vowels and consonants occurred in each string.

c) Accept two strings from the user and determine if the strings are anagrams or not.

6. a) Create a multilevel inheritance for classes vehicle, brand and cost. The vehicle class determines the type of vehicle which is inherited by the class brand which determines the brand of the vehicle. Brand class is inherited by cost class, which tells about the cost of the vehicle. Create another class which calls the constructor of cost class and method that displays the total vehicle information from the attributes available in the super classes.

b) Create an inheritance hierarchy of Figure_3D, Cylinder, Cone, Sphere etc. In the base class and provide methods that are common to all Figure_3Ds and override these in the derived classes to perform different behaviors, depending on the specific type of Figure_3D. Create an array of Figure_3D, fill it with different specific types of Figure_3Ds and call your base class methods.

7. a) Design a package to contain the class Student that contains data members such as name, roll number and another package contains the interface Sports which contains some sports information. Import these two packages in a package called Report which process both Student and Sport and give the report.

b) Write a program that accepts values of different data types and convert them to corresponding wrapper classes and display using the vector

8. a) Write a program to generate a set of random numbers between two numbers x1 and x2, and x1>0.

b) Write a program to implement a new ArrayList class. It should contain add(), get(), remove(), size() methods. Use dynamic array logic.

c) Create an employee class containing at least 3 details along with Id, setters and getters. Insert the employee objects dynamically key as employee id and value as it's corresponding object into a HashMap. Perform Id based search operation on the HashMap.

9. a) Write a program that reads file name from the user then displays information about that file, also read the contents from the file in byte stream to count number of alphabets, numeric values and special symbols. Write these statistics into another file using byte streams.

b) Write a program that reads a CSV file containing a super market data containing product ID, Name, Cost and Quantity of sales and calculate the total revenue of the super market also sort the products in the order of their demand.

c) Write a program that reads a text file containing some technical content and identify the technical terms and sort them alphabetically.
Note: use a file containing stop words (general English and Grammar terms as many terms as possible)

10. a) Write a program that reads two numbers from the user to perform integer division into Num1 and Num2 variables. The division of Num1 and Num2 is displayed if they are integers. If Num1 or Num2 were not an integer, the program would throw a Number Format Exception. If Num2 were Zero, the program would throw an Arithmetic Exception.

b) Create a user defined exception.

11. a) Write a program that creates 3 threads by extending the Thread class. First thread displays "Good Morning" every 1 sec, the second thread displays "Hello" every 2 seconds and the third displays Welcome" every 3 seconds. (Repeat the same by implementing Runnable).

b) Write a program to illustrate Thread synchronization.

12. a) Create a JApplet that displays a message which is scrolling from left to right and vice versa

b) Write a program that displays a sample registration page using Swing controls use appropriate layout managers.

c) Write a program for handling mouse events with adapter classes.

13. a) Create an interface containing 3 radio buttons named line, rectangle and oval. Based on the radio button selected, allow user to draw lines, rectangles or ovals as per the locations selected by the user.

b) Write a program to create a Table inside a JFrame.

c) Create an interface that illustrates JFileChooser class and read CSV file containing employee data of various departments and display the records department wise on the interface.

14. a) For program 12) b) check all the fields filled or not, display success dialogue if all fields are filled with the help of ActionListener. Display respective error dialogue if a field is empty.

b) Write a program to create three JSliders where each represents colors RED, GREEN and BLUE. Each slider has a value from 0 to 255. The background color of the applet is set based on the values retrieved from each slider to form a color using the color class constructor. On sliding any slider, the background color of applet changes.

15. Write a program that implements a simple client/server application. The client sends data to a server. The server receives the data, uses it to produce a result, and then sends the result back to the client. The client displays the result on the console. For ex: The data sent from the client is the radius of a circle, and the result produced by the server is the area of the circle.

Case Studies:

- Grading a Multiple-choice Test for students
- Create a Person class containing basic details like Name, Gender, Mobile and Email. Based on that create and manage the objects that are related to student and employee Classes
- Create a package called Banking containing classes and interfaces related to various banking operations such as withdrawal, deposits, loans and insurance etc. Create two classes related to any two specific banks that uses this package.
- Write a program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the+, - ,*, % operations. Add a text field to display the result.
- Write a program that implements simple chat application using GUI

TEXT BOOKS:

1. Herbert Schildt, "*Java The Complete Reference*", 11th Edition, McGrawHill, 2019

REFERENCE BOOKS:

1. Y. Daniel Liang "*Introduction to Java Programming Comprehensive Version*" 10th Edition, Pearson, 2015