SCMS SCHOOL OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS)
Vidya Nagar, Palissery, Karukutty, Ernakulam - 683576, Kerala, India.
www.scmsgroup.org/sset

SCMS SCHOOL OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS)

VIDYA NAGAR, PALISSERY, KARUKUTTY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

2026 Scheme

Syllabus

Semester 1

Master of Computer Applications

Branch Code - MCA

SEMESTER 1 (S1)

Sl.	Slot	Course Code	Course Title	Type	Cat.	Cr.	Hour Structure				Total Hrs	CIA	SEE	Total Marks
							L	T	P	SL				
1	A	26PCMCA101	Mathematical Foundations for Computing	CC	PC	4	45	15	-	60	120	40	60	100
2	B	26PCMCA102	Digital Fundamentals & Computer Architecture	PC	PC	4	45	15	-	60	120	40	60	100
3	C	26PCMCA103	Advanced Data Structures	PC	PC	4	45	15	-	60	120	40	60	100
4	D	26PCMCA104	Advanced Software Engineering	PC	PC	4	45	15	-	60	120	40	60	100
5	E	26PLMCA105	Programming Lab	PCL	PCL	2	-	-	45	15	60	50	50	100
6	F	26PLMCA106	Data Structures Lab	PCL	PCL	2	-	-	45	15	60	50	50	100
7		26SEMCA107	Entrepreneurship & Innovations in Technology	CC	SEC	1	15	-	-	15	30	100	-	100
8			Minimum one MOOC Course is to be completed	MC	MC	2	-	-	-	60	60	-	-	-
TOTAL						**23**	**195**	**60**	**90**	**345**	**690**	**360**	**340**	**700**

Course Code	**26PCMCA101**		
Course Name	**MATHEMATICAL FOUNDATIONS FOR COMPUTING**		
Teaching Hours	L:45 T:15 P:0 SL:60	**Course Credits**	4
Continuous Internal Assessment Marks	40	**Semester End Exam Marks**	60
Exam Hours	3 Hours	**Total Marks**	100
Course Type	CC	**Course Category**	PC
Prerequisites	Nil		

Course Objectives

This course introduces students to some basic mathematical ideas and tools which are at the core of MCA course. It introduces the concepts of graph theory, set theory and statistics.

Course Outcomes (COs)

CO#	CO Statement	Knowledge Level
CO1	Understand mathematical reasoning in order to read, comprehend and construct mathematical arguments	K2
CO2	Count or enumerate objects and solve counting problems and analyze algorithms	K4
CO3	Solve problems in almost every conceivable discipline using graph models	K3
CO4	Solve the linear system of equations and Calculate the eigen values and eigen vectors of matrices	K3
CO5	Apply the principles of correlation and regression in practical problems	K3

Course Articulation Matrix

CO-PO Mapping

CO#	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	3	3	3	–	–	3	-	-	-	-
CO2	3	3	3	3	–	–	3	-	-	-	-
CO3	3	3	3	3	–	–	3	-	-	-	-
CO4	3	3	3	3	–	–	3	-	-	-	-
CO5	3	3	3	3	–	–	3	-	-	-	-
Avg	**3**	**3**	**3**	**3**	**-**	**-**	**3**	**-**	**-**	**-**	**-**

1: Slight (Low), 2: Moderate (Medium), 3: Substantial (High), -: No Correlation,

Assessment Pattern

Bloom's Category	Continuous Assessment Tests		End Semester Examination
	Test 1	Test 2	
Remember (K1)	10	10	10
Understand (K2)	20	20	20
Apply (K3)	20	20	30
Analyse (K4)	-	-	-
Evaluate (K5)	–	–	–
Create (K6)	–	–	–

Mark distribution

Total Marks	Continuous Internal Evaluation (CIE)	End Semester Examination (ESE)	ESE Duration
100	40	60	3 Hours

Continuous Internal Evaluation (CIE) Pattern

Attendance : 8
Continuous Assessment Tests (2 Nos.) : 20
Assignment / Quiz / Mini Project / Seminar : 12
Part A

- **10 compulsory short-answer questions**
- **2 questions from each of the five modules**
- **3 marks each**
- **Total: 30 marks**

Part B

- **10 descriptive questions (2 from each module)**
- Students shall answer **one question from each module** (5 questions in total).
- Each question may contain **a maximum of two sub-divisions**.
- **6 marks each**
- **Total: 30 marks**

Grand Total: 60 Marks
Course Syllabus

Module: 1	**Contact Hours: 10**
Sets, Set Operations, Relations, Classification of relations, Equivalence Relations, Closures of Relations, Matrix Representation of Relations, Partial Ordering, n-ary Relations, Functions.	

Module: 2	**Contact Hours: 10**
Division Algorithm, GCD, Primes, Euclidean Algorithm, Congruences, Properties of Congruences, Solutions of Linear Congruences. First Order Linear Recurrence Relation, Second Order Linear Homogeneous Recurrence Relations with Constant coefficients, Non-Homogeneous Recurrence Relation.	

Module: 3	**Contact Hours: 8**
Graphs and Graph Models, Graph Terminology and Special Types of Graphs, Representing Graphs and Graph Isomorphism, Connectivity, Euler and Hamilton Paths, Shortest-Path Problems, Planar Graphs, Directed Graph, Multigraph, Connected graph, Euler circuit and trail, Planar and Non-planar Graphs.	
Module: 4	**Contact Hours: 8**
Linear system of equations, coefficient matrix, augmented matrix, Gauss elimination method and back substitution, elementary row operations, row equivalent systems, Gauss elimination- three possible cases, Row Echelon form and information from it, Linear independence- rank of a matrix. Solution of linear system, fundamental theorem of non- homogeneous linear system (without proof). Homogeneous linear system (theory only), Matrix eigen value problem- determination of eigen values and eigen vectors, Basis of eigen vectors- diagonalization of matrix- Quadratic form-principle axis theorem	

(without proof).
Module: 5 **Contact Hours:9**
Bivariate data – Scatter Diagram – Interpretation of the nature and degree of relation using scattered diagram - Curve fitting – Principle of least squares – fitting a straight line – fitting a parabola – linear correlation and regression – Karl's Pearson's Coefficient of Correlation – Spearman's rank correlation coefficient (problems based on the formula).

Suggested Assignments for Course Teacher's Assessment (CTA) Through Self-Learning Component:

1. Generate and solve simple problems on **sets and set operations**, and verify the solutions using Venn diagrams.

2. Ask AI to generate examples of different types of **relations** and classify them as reflexive, symmetric, antisymmetric, and transitive.

3. Create examples of **equivalence relations** and independently verify whether the given relations satisfy all required properties.

4. Explore **functions** using AI-generated examples and identify one-one, onto, and bijective functions.

5. Generate problems involving the **Euclidean algorithm and GCD**, solve them manually, and compare the results with AI-generated solutions.

6. Explore real-life applications of **congruences and modular arithmetic**, such as clock arithmetic, coding, and hashing, with the assistance of AI.

7. Generate simple **recurrence relations**, determine the initial terms, identify patterns, and verify the solutions.

8. Model real-life systems such as **computer networks, social networks, or road networks as graphs**.

9. Generate small graphs using AI and investigate **Euler paths, Euler circuits, Hamilton paths, and Hamilton cycles**.

10. Create a weighted network and determine the **shortest path** between two vertices; verify the result using Dijkstra's algorithm.

11. Use AI-generated pairs of graphs to explore **graph isomorphism** by comparing degree sequences and possible vertex mappings.

12. Use AI to generate small matrices and explore the calculation of **eigenvalues and eigenvectors**, verifying the results independently.

13. Explain the concept of **matrix diagonalization** through a simple numerical example and verify the conditions manually.

14. Generate a small dataset and create a **scatter diagram**, followed by interpretation of the nature and degree of relationship.

15. Use AI and spreadsheet tools to perform **curve fitting using the principle of least squares** for a given dataset.

Text Books

1. David M. Burton, "Elementary Number Theory", McGraw-Hill, 7th Edition (2012).

2. Ralph P Grimaldi, "Discrete and Computational Mathematics: An applied introduction", Pearson Education, 5th Edition, (2007).

3. Erwin Kreyszig, "Advanced Engineering Mathematics", 10th ed., Wiley.

4. Gupta S.C and Kapoor V .K, "Fundamentals of Mathematical Statistics", Sultan Chand and Sons 11th edition.

Reference Books and Extended Learning

1. C. Liu, "Elements of Discrete Mathematics: A Computer Oriented Approach", McGraw-Hill, 4th Edition (2012).

2. Jean-Paul Tremblay ,"Discrete Mathematical Structures with applications to Computer science", McGraw-Hill, 1st Edition (2001).

3. Kenneth H. Rosen , "Discrete mathematics and its applications", McGraw-Hill, (7th Edition), (Smartbook available).

4. Marty Lewinter, Jeanine Meyer, "Elementary Number Theory with Programming", Wiley-Blackwell (2015).

5. David S. Moore and George P. McCabe, "Introduction to practice of statistics", W.H. Freeman & Company, 5th Edition (2005).

6. Douglas C. Montgomery and George C. Runger, "Applied Statistics and Probability for Engineers", Wiley India, 5th Edition (2012).

7. Veerarajan T, "Probability and Random Process", 3rd Edition, Tata McGraw-Hill (2002)

8. G. Jay Kerns, "Introduction to Probability and Statistics Using R", Chapman & Hall (2010).

9. B.S Grewal. Higher Engineering Mathematics, Khanna Publishers, New Delhi.

Course Code	**26PCMCA102**		
Course Name	**DIGITAL FUNDAMENTALS & COMPUTER ARCHITECTURE**		
Teaching Hours	L:45 T:15 P:0 SL:60	**Course Credits**	4
Continuous Internal Assessment Marks	40	**Semester End Exam Marks**	60
Exam Hours	3 Hours	**Total Marks**	100
Course Type	PC	**Course Category**	PC
Prerequisites	Nil		

Course Objectives

The primary aim of this course is to understand the fundamentals behind the digital logic design and gain the experience to design digital circuits and systems. Students should also acquire some understanding and appreciation of a computer system's functional components, their characteristics, performance and interactions. They need to understand the computer architecture in order to make best use of the software tools and computer languages they use to create programs.

Course Outcomes (COs)

CO#	CO Statement	Knowledge Level
CO1	Apply the basics of digital electronics to design and realize simple combinational logic circuits	K3
CO2	Apply the digital electronics principles to design sequential logic circuits.	K3
CO3	Understand the different design features of computer architecture, Five key components of a computer, processor and memory making technologies, addressing modes & instruction formats.	K2
CO4	Understand Processor logic design conventions and data path, pipelining and hazards, I/O organization, Interrupts and direct memory access	K2
CO5	Understand and different types of memories - RAM, ROM, Cache memory, virtual memory etc. Apply the different memory design techniques.	K2
CO6	Understand the concept of single board computers like Arduino, Raspberry Pi etc. and apply the same in practical applications.	K2

Course Articulation Matrix
CO-PO Mapping

CO#	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	3	2	1	–	–	1	–	–	–	–
CO2	3	3	2	1	–	–	1	–	–	–	–
CO3	1	1	-	1	–	–	1	–	–	–	–
CO4	1	1	-	1	–	–	1	–	–	–	–
CO5	2	2	1	-	–	–	1	–	–	–	–
CO6	1	1	2	1	2	–	2	-	-	–	-
Avg	**1.8**	**1.8**	**1.8**	**1**	**2**	**-**	**1.2**	**-**	**-**	**-**	**-**

1: Slight (Low), 2: Moderate (Medium), 3: Substantial (High), -: No Correlation,

Assessment Pattern

Bloom's Category	Continuous Assessment Tests		End Semester Examination
	Test 1	Test 2	
Remember (K1)	10	10	20
Understand (K2)	20	20	20
Apply (K3)	20	20	20
Analyse (K4)	–	–	–
Evaluate (K5)	–	–	–
Create (K6)	–	–	–

Mark distribution

Total Marks	Continuous Internal Evaluation (CIE)	End Semester Examination (ESE)	ESE Duration
100	40	60	3 Hours

Continuous Internal Evaluation (CIE) Pattern
Attendance : 8
Continuous Assessment Tests (2 Nos.) : 20
Assignment / Quiz / Mini Project / Seminar : 12

Part A

- **10 compulsory short-answer questions**
- **2 questions from each of the five modules**
- **3 marks each**
- **Total: 30 marks**

Part B

- **10 descriptive questions (2 from each module)**
- Students shall answer **one question from each module** (5 questions in total).
- Each question may contain **a maximum of two sub-divisions**.
- **6 marks each**
- **Total: 30 marks**

Grand Total: 60 Marks
Course Syllabus

Module: 1	**Contact Hours: 11**
Representation of signed numbers – 1's complement and 2's complement ,Logic gates – AND - OR – NOT - NAND- NOR - XOR , Boolean algebra - Basic laws and theorems , Boolean functions- truth table, Standard forms of Boolean Expressions – Sum of Products and Product of Sums - minimization of Boolean function using Karnaugh map method - Realization using logic gates, Floating point numbers. Combinational Circuits - Half adder - Full Adder- Decoder -Encoder- Multiplexer – Demultiplexer	
Module: 2	**Contact Hours: 10**
Sequential circuit - Clocking, Flip flops - SR – JK- D -T flip flops, Counters - Synchronous and asynchronous counters - UP/DOWN counters , Registers - Serial in serial out - Serial in parallel out - Parallel in serial out - Parallel in parallel out registers	
Module: 3	**Contact Hours: 10**
Computer abstractions and technology - Introduction, Computer architecture -8 Design features, Application program - layers of abstraction, Five key components of a computer, Technologies for building processors and memory, Performance, Instruction set principles – Introduction, Classifying instruction set architectures, Memory addressing, Encoding an instruction set.	
Module: 4	**Contact Hours: 9**

The Processor - Introduction, Logic design conventions, Building a datapath, A simple implementation scheme, An overview of pipelining - Pipelined datapath and control - Structural hazards - Data hazards - Control hazards I/O organization - Accessing I/O devices, interrupts - handling multiple devices, Direct memory access	
Module: 5	**Contact Hours: 8**
The Memory System – basic concepts, semiconductor RAM memories - organization – static and dynamic RAM, Structure of larger memories, semiconductor ROM memories, Speed, Size and cost ,Cache memory – mapping functions – replacement algorithms , Virtual memory – paging and segmentation.	

Suggested Assignments for Course Teacher's Assessment (CTA) Through Self-Learning Component:

Assignment 1: Boolean Function Minimization & Combinational Circuit Design

Focus: Boolean Algebra, Truth Tables, SOP/POS, Karnaugh Maps, Logic Gates, Combinational Circuits

Objective: Understand Boolean function representation and minimization, and apply K-map techniques to design optimized combinational circuits.

Tasks:

1. **Boolean Function Analysis:** Given 3–4 Boolean functions, construct the truth tables and derive their **SOP and POS** representations. Simplify the functions using Boolean algebra.
2. **K-Map Minimization:** Minimize the given Boolean functions using **3-variable and 4-variable Karnaugh maps**. Show the grouping process and obtain the minimized expressions.
3. **Circuit Realization:** Implement the minimized Boolean functions using basic gates and universal **NAND/NOR gates**. Simulate the circuits using **Logisim Evolution** and compare the original and optimized designs.

Deliverable: A technical report (4–5 pages) containing truth tables, Boolean expressions, K-map solutions, circuit diagrams, simulation screenshots, and a comparison of the original and optimized circuits.

Assignment 2: Processor Pipelining & Hazard Analysis

Focus: Datapath, Pipelining, Structural Hazards, Data Hazards, Control Hazards

Objective: Understand instruction pipelining and analyze how different types of hazards affect processor performance.

Tasks:

1. **Pipeline Walkthrough:** Given a sequence of 8–10 instructions, trace their execution through a **5-stage pipeline** consisting of IF, ID, EX, MEM, and WB stages.
2. **Hazard Identification:** Identify **structural, data, and control hazards** in the instruction sequence and represent their effects using pipeline timing diagrams.
3. **Hazard Resolution & Performance:** Apply techniques such as **forwarding, stalling, and branch prediction** to resolve the hazards. Calculate the execution time and speedup before and after hazard resolution.

Deliverable: A technical report (4–5 pages) containing pipeline diagrams, hazard analysis, timing diagrams, calculations, and a critical discussion of pipeline performance.

Assignment 3: Cache Mapping & Memory Performance Analysis

Focus: Cache Memory, Direct Mapping, Associative Mapping, Set-Associative Mapping, Cache Hits/Misses

Objective: Understand cache organization and analyze how different cache mapping techniques influence memory access performance.

Tasks:

1. **Mapping Walkthrough:** Given a main-memory and cache configuration, manually determine the cache location of a sequence of memory addresses using **direct, fully associative, and set-associative mapping**.
2. **Hit/Miss Analysis:** Calculate the number of cache hits and misses for a given memory-access sequence. Determine the **hit ratio, miss ratio, and average memory access time**.
3. **Simulation & Comparison:** Implement the three cache mapping techniques in Python. Vary cache size and block size, plot the **cache hit ratio**, and analyze the performance differences among the mapping techniques.

Deliverable: A technical report (4–5 pages) containing address-mapping calculations, cache diagrams, Python implementation, performance plots, and a critical comparison of the three cache mapping techniques.

Text Books

1. Floyd, "***Digital Fundamentals***", Pearson Education, 10th Edition (2011).(Module 1 & 2)
2. J. Hennessy and D. Patterson, "***Computer Organization and Design: The Hardware/Software Interface***", 5th Edition. (Module 3 & 4)
3. J. Hennessy and D. Patterson, "***Computer Architecture, A quantitative approach***", 5th Edition. (Module 3)
4. Hamacher, Vranesic & Zaky, "***Computer Organization***" (5th Ed), McGraw Hill. (Module 4 & 5)

Reference Books and Extended Learning

1. William Stallings, "***Computer Organization and Architecture: Designing for Performance***", Pearson, 9/e, 2013.
2. R.P.Jain ,"***Modern Digital Electronics***", McGraw Hill.,Fourth Edition,2009
3. Mano, "***Digital Design : With an Introduction to Verilog HDL***", Pearson Education, 5th Edition (2014)

Course Code	**26PCMCA103**		
Course Name	**ADVANCED DATA STRUCTURES**		
Teaching Hours	L:45 T:15 P:0 SL:60	**Course Credits**	4
Continuous Internal Assessment Marks	40	**Semester End Exam Marks**	60
Exam Hours	3 Hours	**Total Marks**	100
Course Type	PC	**Course Category**	PC
Prerequisites	Basic Data Structures		

Course Objectives

A graduate course in Computer Applications should give due exposure to the recent developments. Since Data structures is a central pillar of any program on Computer Science/ Applications, this course is designed to build upon the knowledge acquired at the undergraduate level and familiarize students with a bunch of modern data structures which are quite useful to solve, in the most effective manner, the modern, real life problems.

Course Outcomes (COs)

CO#	CO Statement	Knowledge Level
CO1	Remember the Basic Data Structures and understand the Set Data Structure and its implementation.	K1
CO2	Understand Advanced Tree Structures for the design of efficient algorithms	K2
CO3	Understand Advanced Heap Structures suitable for solving Computational problems involving Optimization and analyzing these data structures using amortized analysis.	K2
CO4	Understand Advanced Graph algorithms suitable for solving advanced computational problems	K2
CO5	Understand the basic operation of Block chaining along with the data structures used in it and the challenges in Blockchain data.	K2

Course Articulation Matrix

CO-PO Mapping

CO#	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	P010	PO11
CO1	3	2	2	–	1	-		-	-	-	-
CO2	2	2	3	2	1	-	1	-	-	-	-
CO3	2	3	3	2	1	-	1	-	-	-	-
CO4	3	3	2	1	2	-	1	-	-	-	-

CO5	3	2	2	2	3	-	1	-	-	-	-
Avg	**2.6**	**2.4**	**2.4**	**1.8**	**1.6**	**-**	**1**	**-**	**-**	**-**	**-**

1: Slight (Low), 2: Moderate (Medium), 3: Substantial (High), -: No Correlation,

Assessment Pattern

Bloom's Category	Continuous Assessment Tests		End Semester Examination
	Test 1	Test 2	
Remember (K1)	10	10	10
Understand (K2)	20	20	20
Apply (K3)	20	20	30
Analyse (K4)	–	–	–
Evaluate (K5)	–	–	–
Create (K6)	–	–	–

Mark distribution

Total Marks	Continuous Internal Evaluation (CIE)	End Semester Examination (ESE)	ESE Duration
100	40	60	3 Hours

Continuous Internal Evaluation (CIE) Pattern
Attendance : 8
Continuous Assessment Tests (2 Nos.) : 20
Assignment / Quiz / Mini Project / Seminar : 12
Part A

- **10 compulsory short-answer questions**
- **2 questions from each of the five modules**
- **3 marks each**
- **Total: 30 marks**

Part B

- **10 descriptive questions (2 from each module)**
- Students shall answer **one question from each module** (5 questions in total).
- Each question may contain **a maximum of two sub-divisions**.
- **6 marks each**
- **Total: 30 marks**

Grand Total: 60 Marks
Course Syllabus

Module: 1	**Contact Hours: 12**
Review of basic data structures- Array, linked list and its variants, Stack ,Queue and Trees Set Data Structure:- Representation of sets, set implementation using bit string. Hashing :- Simple hash functions, Collision and Collision Resolution techniques. Amortised Analysis - Aggregate, Accounting and Potential Methods (using the examples Multipop Stack and Incrementing Binary Counter only) Disjoint sets- representations, Union, Find algorithms	
Module: 2	**Contact Hours: 10**
Advanced Tree Structures:- Balanced Binary Search trees, Red-Black trees- Properties of Red Black trees, Rotations, Insertion, Deletion. B-Trees- Basic operations on B-Trees – Insertion and Deletion, Introduction to Splay Trees and Suffix Trees	
Module: 3	**Contact Hours: 10**
Advanced Heap Structures:- Mergeable Heaps and operations on Mergeable Heaps. Binomial Heaps, Binomial Heap operations and Analysis, Fibonacci Heaps, Fibonacci Heap operations and Analysis**.**	
Module: 4	**Contact Hours: 10**
Advanced Graph Structures : Representation of graphs, Depth First and Breadth First Traversals, Topological Sorting, strongly connected Components and Biconnected Components Minimum Cost Spanning Tree algorithms- Prim's Algorithm, Kruskal' Algorithm,. Shortest Path Finding algorithms – Dijikstra's single source shortest paths algorithm	
Module: 5	**Contact Hours: 8**
Blockchain Data Structure:- Blockchain Architecture, Blockchain Data Structures and Data types, Contract Data, Problems to be solved in Blockchain data analysis	

Suggested Assignments for Course Teacher's Assessment (CTA) Through Self-Learning Component:

Assignment 1: Red-Black Trees — Mechanics, Invariants & Operations

Focus: Balanced Binary Search Trees, Red-Black Trees (Properties, Rotations, Insertion, Deletion)

- **Objective:** Understand how coloring rules and rotations enforce an $O(\log n)$ height bound and master the trace of complex structural repairs during insertion and deletion.
- **Tasks:**
    1. **Invariant Analysis & Proofs:** Prove mathematically why a Red-Black Tree with n internal nodes has a maximum height of $2 \log_2(n + 1)$. Explain how the 5 fundamental properties prevent severe imbalance.
    2. **Step-by-Step Step Execution (Hand Tracing):**

- Starting with an empty Red-Black tree, insert keys sequentially: `[10, 20, 30, 15, 25, 5, 1, 18]`. Draw the tree after every insertion, explicitly identifying recoloring cases and tree rotations (Left-Rotate, Right-Rotate).
- From the resulting tree, delete keys `[10, 30]` step-by-step. Document every deletion case (e.g., handling "double black" nodes) and state how balance is restored.

3. **Implementation:** Write a standard Red-Black Tree in C++, Java, or Python that includes a visual tree-printer or explicit validation function that verifies all Red-Black invariants after every operation.

- **Deliverable:** A technical report containing hand-drawn or diagrammed state transitions for insertion/deletion, the structural proof, and working source code.

Assignment 2: External Memory Indexing with B-Trees

Focus: B-Trees, Multi-way Search Trees, Node Splitting and Merging, Insertion & Deletion Mechanics

- **Objective:** Master multi-way search trees used in disk-based storage engines, database indexing, and file systems.
- **Tasks:**
  1. **Tracing B-Tree Operations:** Given a B-Tree of Minimum Degree $t = 3$ (Order 5, max keys = 4, min keys = 2):
     - Perform step-by-step insertions of: `[5, 15, 25, 35, 45, 55, 65, 20, 30, 40]`. Show node splits (proactive vs. reactive) and root ascensions.
     - Perform step-by-step deletions of keys: `[35, 5]` involving borrowing from siblings (redistribution) and merging of nodes underflowed below $t - 1$ keys.
  2. **Disk I/O Analysis:** Calculate the maximum theoretical height of a B-Tree containing 10^7 keys with block size 4 KB and key size 32 bytes. Contrast its height and expected disk I/O operations with a Red-Black Tree storing the same number of keys.
- **Deliverable:** Detailed execution trace diagrams for B-Tree insertions/deletions, alongside the analytical calculation of disk I/O savings.

Assignment 3: Self-Adjusting & Specialized Trees — Splay Trees & Suffix Trees

Focus: Splay Trees (Amortized Analysis, Splay Operations), Suffix Trees (Pattern Matching & Applications)

- **Objective:** Explore dynamically self-adjusting search trees for caching/frequent-access scenarios, and specialized string-indexing structures.
- **Tasks:**
  1. **Splay Tree Mechanics:**
     - Explain the **Zig**, **Zig-Zig**, and **Zig-Zag** splay operations.
     - Trace the search for keys `[12, 3]` on a given BST using splay mechanics. Show how accessing elements brings them to the root and reduces path lengths for subsequent accesses.
     - Discuss the difference between worst-case single-operation complexity and **amortized time complexity** $O(\log n)$ for Splay Trees.
  2. **Suffix Tree Construction & String Applications:**
     - Construct a **Suffix Tree** for the text string `T = "BANANA$"` using Ukkonen's algorithm concepts or implicit suffix trie compaction.

- Demonstrate how the constructed Suffix Tree solves the following problems in $O(m)$ time (where m is pattern length):
  - Exact pattern search for $P = \text{"ANA"}$.
  - Finding the Longest Repeated Substring in T.
- **Deliverable:** Visual traces of Splay Tree operations, drawn Suffix Tree diagrams, and a conceptual synthesis on when to choose Splay Trees over Red-Black Trees

Text Books

1. Cormen T.H., Leiserson C.E, Rivest R.L. and Stein C, ***Introduction to Algorithms,*** Prentice Hall India, New Delhi, 2004 [Modules 1 to 4]

Yang, Xiaojing, Jinshan Liu, and Xiaohe Li. "***Research and Analysis of Blockchain Data." Journal of Physics: Conference Series. Vol. 1237***. No. 2. IOP Publishing, 2019.

Reference Books and Extended Learning

1. Kleinberg, Jon, and Eva Tardos. ***Algorithm design***. Pearson Education India, 2006.

2. Aho A.V., Hopcroft J.E., and Ullman J.D., ***Data Structures and Algorithms***, Pearson Education, New Delhi, 1983.

3. Sahni S., ***Data Structures, Algorithms, and Applications in C++,*** Mc Graw Hill, Singapore, 1998.

Course Code	**26PCMCA104**		
Course Name	**ADVANCED SOFTWARE ENGINEERING**		
Teaching Hours	L:45 T:15 P:0 SL:60	**Course Credits**	4
Continuous Internal Assessment Marks	40	**Semester End Exam Marks**	60
Exam Hours	3 Hours	**Total Marks**	100
Course Type	PC	**Course Category**	PC
Prerequisites	Programming proficiency in at least one of C, C++, Java, Python or PHP programming languages.		

Course Objectives

Most of the programs on Computer Applications do not give due importance to teach Software Engineering in an Industry perspective. But this course, built upon the tools and techniques prevalent in

Industry today, is supposed to make students Industry-ready.

Course Outcomes (COs)

CO#	CO Statement	Knowledge Level
CO1	Get a full view of the Software life cycle	K2
CO2	Gain a deep knowledge of Software Planning, Analysis and Design and Software Engineering Models	K2
CO3	Have a great comprehension of Coding Practices, Version Control using 'git' and Software Quality	K2
CO4	Acquire ample grasp of Design Patterns	K2
CO5	Get deeply familiarized with Software Testing and its automation	K2
CO6	Start using Agile Methodology	K3
CO7	Begin to apply CI/CD techniques in Software development	K3

Course Articulation Matrix
CO-PO Mapping

CO#	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	-	2	2	-	-	-		3	-	-	1
CO2	-	3	3	-	-	-	-	3	-	-	-
CO3	-	-	-	-	3	-	-	-	3	2	2
CO4	-	-	3	-	3	-	-	-	-	-	-
CO5	-	-	-	-	3	-	-	-	-	2	3
CO6	-	-	-	-	2	-	-	2	2	-	2

CO7	-	-	-	-	3	-	-	1	-	2	-
Avg	**-**	**2.5**	**2.7**	**-**	**2.8**	**-**	**-**	**2.2**	**2.5**	**2**	**2**

1: Slight (Low), 2: Moderate (Medium), 3: Substantial (High), -: No Correlation,

Assessment Pattern

Bloom's Category	Continuous Assessment Tests		End Semester Examination
	Test 1	Test 2	
Remember (K1)	10	10	10
Understand (K2)	20	20	20
Apply (K3)	10	10	20
Analyse (K4)	-	-	-
Evaluate (K5)	–	–	–
Create (K6)	10	10	10

Mark distribution

Total Marks	Continuous Internal Evaluation (CIE)	End Semester Examination (ESE)	ESE Duration
100	40	60	3 Hours

Continuous Internal Evaluation (CIE) Pattern
Attendance : 8
Continuous Assessment Tests (2 Nos.) : 20
Assignment / Quiz / Mini Project / Seminar : 12
Part A

- **10 compulsory short-answer questions**
- **2 questions from each of the five modules**
- **3 marks each**
- **Total: 30 marks**

Part B

- **10 descriptive questions (2 from each module)**
- Students shall answer **one question from each module** (5 questions in total).
- Each question may contain **a maximum of two sub-divisions**.
- **6 marks each**
- **Total: 30 marks**

Grand Total: 60 Marks
Course Syllabus

Module: 1	**Contact Hours: 8**

Introduction to Software Engineering: What is Software Engineering, Characteristics of Software. Life cycle of a software system: software design, development, testing, deployment, Maintenance. Project planning phase: project objectives, scope of the software system, empirical estimation models, COCOMO, staffing and personnel planning. Software Engineering models: Predictive software engineering models, model approaches, prerequisites, predictive and adaptive waterfall, waterfall with feedback (Sashimi), incremental waterfall, V model; Prototyping and prototyping models. Software requirements specification, Eliciting Software requirements, Requirement specifications, Software requirements engineering concepts, Requirements modelling, Requirements documentation. Use cases and User stories.

Module: 2	**Contact Hours: 10**

Programming Style Guides and Coding Standards; Literate programming and Software documentation; Documentation generators, Javadoc, phpDocumentor. Version control systems basic concepts; Concept of Distributed version control system and Git; Setting up Git; Core operations in Git version control system using command line interface (CLI): Clone a repository; View history; Modifying files; Branching; Push changes, Clone operation, add, commit, log, diff commands, conflict resolution.

Pushing changes to the master; Using Git in IDEs and UI based tools. Software Quality: Understanding and ensuring requirements specification quality, design quality, quality in software development, conformance quality.

Module: 3	**Contact Hours: 10**

OOP Concepts; Design Patterns: Basic concepts of Design patterns, how to select a design pattern, Creational patterns, Structural patterns, Behavioral patterns. Concept of Anti-patterns. Unit testing and Unit Testing frameworks, the xUnit Architecture, Writing Unit Tests using at least one of Junit (for Java), unittest (for Python) or phpdbg (PHP). Writing tests with Assertions, defining and using Custom Assertions, single condition tests, testing for expected errors, Abstract test.

Module: 4	**Contact Hours: 10**

Concepts of Agile Development methodology; Scrum Framework. Software testing principles, Program inspections, Program walkthroughs, Program reviews; Blackbox testing: Equivalence class testing, Boundary value testing, Decision table testing, Pairwise testing, State transition testing, Use-case testing; White box testing: control flow testing, Data flow testing. Testing automation: Defect life cycle; Regression testing, Testing automation; Testing non-functional requirements.

Module: 5	**Contact Hours:10**

Software Configuration Management: Using version control, managing dependencies, managing

software configuration, Managing build and deployment environments. Continuous Integration: Prerequisites for continuous integration, Essential practices. Continuous Delivery: Principles of Software delivery, Introduction and concepts. Build and deployment automation, learn to use Ansible for configuration management. Test automation (as part of continuous integration), Learn to set up test automation cases using Robot Framework.

Suggested Assignments for Course Teacher's Assessment (CTA) Through Self-Learning Component:

Assignment 1: Software Requirements Engineering & Process Model Analysis
Focus: Software Development Life Cycle, Predictive/Adaptive Models, Waterfall, V-Model, Prototyping, Requirements Engineering, Use Cases, User Stories
· **Objective:** Understand software development life-cycle models and apply requirements engineering techniques to analyze and document software requirements.
· **Tasks:**

1. **Process Model Analysis:** Select a real-world software system and analyze its development using **Waterfall, V-Model, Incremental, and Prototyping models**. Compare the models based on flexibility, customer involvement, risk, testing, and suitability.
2. **Requirements Elicitation & Modelling:** Identify functional and non-functional requirements for the selected system. Prepare a **Software Requirements Specification (SRS)** containing system scope, objectives, assumptions, constraints, and requirements.
3. **Use Cases & User Stories:** Develop at least **5 use cases and 8 user stories** for the selected system. Map the user stories to functional requirements and identify suitable acceptance criteria.

· **Deliverable:** A technical report (5–6 pages) containing SDLC model comparison, SRS document, requirements classification, use-case diagrams, user stories, and a critical discussion on the suitability of different development models.
Assignment 2: Git-Based Software Development & Unit Testing
Focus: Coding Standards, Software Documentation, Git, Version Control, OOP, Design Patterns, Unit Testing
· **Objective:** Apply professional software development practices using Git-based version control, design patterns, documentation, and automated unit testing.
· **Tasks:**

1. **Git Workflow Implementation:** Create a Git repository for a small OOP-based application. Demonstrate **clone, add, commit, log, diff, branch, merge, push, and conflict resolution** operations using the command line or an IDE.
2. **Design & Documentation:** Apply at least **two suitable design patterns** such as Singleton, Factory, Observer, Strategy, or Adapter. Document the selected patterns, their purpose, class relationships, and advantages. Generate code documentation using **Javadoc or an equivalent documentation tool**.
3. **Unit Testing:** Develop unit tests using **JUnit or Python unittest**. Include assertions, boundary conditions, expected errors/exceptions, and custom assertions where appropriate. Measure test coverage and analyze the identified defects.

· **Deliverable:** A technical report (5–6 pages) containing the Git repository link, commit/branch history screenshots, class/design diagrams, design-pattern implementation, generated documentation, unit-test cases, test results, and coverage analysis.

Assignment 3: Agile Testing, CI/CD & Test Automation

Focus: Agile/Scrum, Software Testing, Black-Box Testing, White-Box Testing, Continuous Integration, Continuous Delivery, Ansible, Robot Framework

· **Objective:** Understand modern Agile software development and implement an automated testing and CI/CD workflow for a software application.

· **Tasks:**

1. **Agile Testing Strategy:** Select a small software application and prepare a **Scrum-based development plan** containing user stories, product backlog, sprint planning, and acceptance criteria. Design test cases using **equivalence partitioning, boundary value analysis, decision tables, and state-transition testing**.
2. **Test Automation:** Implement automated functional test cases using **Robot Framework**. Include positive, negative, boundary, and regression test cases. Record test execution results and analyze the defects identified.
3. **CI/CD & Deployment Automation:** Set up a basic **Continuous Integration/Continuous Delivery pipeline** using a suitable platform such as GitHub Actions or Jenkins. Integrate automated tests into the pipeline and use **Ansible** to automate a simple application configuration or deployment task.

· **Deliverable:** A technical report (5–6 pages) containing the Scrum plan, test strategy, test cases, Robot Framework scripts, test execution results, CI/CD pipeline configuration, Ansible playbook, screenshots, and analysis of the automated development workflow.

Text Books

Reference Books and Extended Learning

1. Philip A. Laplante, *What Every Engineer Should Know about Software Engineering*, CRC Press [Module 1]
2. Murali Chemuturi, *Mastering Software Quality Assurance: Best Practices, Tools and Technique for Software Developers*, J Ross Publishing [Module 2]
3. Ben Straub, Scott Chacon, *Pro Git*, 2nd Edition, Apress [Module 2]
4. Erich Gamma et. al., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley [Moule 3]
5. Vaskaran Sarcar, *Java Design Patterns: A Hands-On Experience with Real-World Examples*, Apress [Module 3]
6. Alistair Cockburn and Robert Cecil Martin, *Agile Software Development: The Cooperative Game (2nd edition)*, Addition Wesley [Module 4]
7. Ken Schwaber , *Agile Software Development with Scrum*, Pearson [Module 4]
8. Lisa Crispin, Agile Testing: *A Practical Guide for Testers and Agile Teams*, Adison Wesley
9. Paul Hamill, *Unit Test Frameworks*, O'Reilly Media [Module 4]
10. Glenford J. Myers, et. al., *The Art of Software Testing*, Wiley [Module 4, 5]
11. Lee Copeland, *A Practitioner's Guide to Software Test Design*, Artech House Publishers

[Module 4, 5]
12. Jez Humble and David Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation,* Pearson Education [Module 5]

Course Code	**26PLMCA105**		
Course Name	**PROGRAMMING LAB**		
Teaching Hours	L:0 T:0 P:45 SL:15	**Course Credits**	2
Continuous Internal Assessment Marks	50	**Semester End Exam Marks**	50
Exam Hours	3 Hours	**Total Marks**	100
Course Type	PCL	**Course Category**	PCL
Prerequisites	Nil		

Course Objectives

This course introduces a basic step towards program writing and develops the logical ability and problem-solving skill using Python Programming Language. Students are able to do testing and debugging of code written in Python.

Course Outcomes (COs)

CO#	CO Statement	Knowledge Level
CO1	Understands basics of Python Programming language including input/output functions, operators, basic and collection data types	K2
CO2	Implement decision making, looping constructs and functions	K3

CO3	Design modules and packages - built in and user defined packages	K6
CO4	Implement object-oriented programming and exception handling.	K3
CO5	Create files and form regular expressions for effective search operations on strings and files.	K6

Course Articulation Matrix
CO-PO Mapping

CO#	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	2	2	2	1	2	–	–	–	–	–	–
CO2	3	3	3	2	2	–	–	–	–	–	–
CO3	3	3	3	3	3	–	–	–	–	–	1
CO4	3	3	3	3	3	–	–	–	–	–	1
CO5	3	3	3	3	3	–	–	–	–	–	1
Avg	**2.8**	**2.8**	**2.8**	**2.5**	**2.7**	–	–	–	–	–	**1**

1: Slight (Low), 2: Moderate (Medium), 3: Substantial (High), -: No Correlation,

Assessment Pattern

Bloom's Category	Continuous Assessment Tests		End Semester Examination
	Test 1	Test 2	
Remember (K1)			
Understand (K2)			
Apply (K3)	10	10	10
Analyse (K4)	10	10	10
Evaluate (K5)	10	10	10
Create (K6)	20	20	20

Mark distribution

Total Marks	Continuous Internal Evaluation (CIE)	End Semester Examination (ESE)	ESE Duration
100	50	50	3 Hours

Continuous Internal Evaluation (CIE) Pattern

Maximum Marks: 50	
Attendance	15%
Maintenance of daily lab record and GitHub management	20%
Regular class viva	15%
Timely completion of day to day tasks	20%
Tests/Evaluation	30%

End Semester Examination Pattern

Maximum Marks: 50			
Verification of Daily program record and Git Repository			5 marks
Viva			10 marks
Problem solving (Based on difficulty level, one or more questions may be given)	Flowchart / Algorithm / Structured description of problem to explain how the problem can be solved / Interface Design	15%	35 marks
	Program correctness	50%	
	Code efficiency	15%	
	Formatted output and Pushing to remote Git repository	20%	
Total Marks			50 marks

Course Contents and Lab Schedule

Topic	No. of hours
1. Input, Output and Import Functions	3
2. Operators	5
3. Data Types	6
4. Decision Making & Loops	6
5. Functions	5
6. Modules and Packages	6
7. File Handling	5
8. Object Handling	5
9. Exception Handling	2
10. Regular Expressions	4

Suggested Assignments for Course Teacher's Assessment (CTA) Through Self-Learning:

Assignment 1: Data Type Precision, Limits & Numerical Computing

Focus: Primitive Data Types, Memory Representation, Type Casting, Overflow/Underflow, Floating-Point Precision Issues

- **Objective:** Understand how computers represent integers and floating-point numbers in memory, the limitations of standard data types, and how to prevent numerical errors.
- **Tasks:**
    1. **Floating-Point Diagnostics:**
        - Write a program to explore floating-point precision issues (e.g., demonstrating why `0.1 + 0.2 != 0.3` in standard IEEE 754 floating-point representation).

- Implement an iterative summation loop using both standard 32-bit floats and 64-bit doubles to calculate a large converging series (e.g., Leibniz formula for π or Harmonic series) and measure precision degradation over 10^6 iterations.

2. **Overflow & Safe Arithmetic:**
   - Demonstrate integer overflow/underflow scenarios using minimum and maximum bounds of standard signed/unsigned integer types.
   - Implement safe arithmetic functions (`safe_add`, `safe_multiply`) that perform conditional checks before execution to prevent arithmetic overflow.
3. **Custom Bitwise Converter:** Implement a tool using bitwise operations to extract and display the sign bit, exponent, and mantissa/fraction bits of an input floating-point number.

- **Deliverable:** A technical report explaining memory representations, code demonstration of edge cases, and an analysis of floating-point arithmetic hazards.

Assignment 2: Rule-Based Decision Systems & Flow Control

Focus: Decision-Making Structures (`if-else`, `switch-case` / `match-case`), Nested Conditions, Boolean Logic Optimization

- **Objective:** Construct complex decision trees and conditional systems while writing clean, readable, and computationally efficient branch logic.
- **Tasks:**
  1. **Financial Tax & Dynamic Fee Engine:**
     - Design a complete decision-making program that computes progressive income tax based on income slabs, tax regimes (e.g., Old vs. New tax regimes), age brackets, and deduction criteria.
     - Ensure the program handles edge cases using logical operators (`AND`, `OR`, `NOT`) without redundant conditional checks.
  2. **Logic Simplification & Refactoring:**
     - Take a provided complex nested `if-else` decision structure (or write one representing a system like a medical triage or credit approval engine) and refactor it into:
       - A simplified boolean expression.
       - A multi-branch `switch-case` or lookup-driven conditional structure.
  3. **Validation & Defensive Programming:** Build a robust user input validator using conditional statements that screens for invalid data types, out-of-bounds numerical inputs, and malformed characters before processing.
- **Deliverable:** A fully commented source code package containing the tax engine, a flow diagram of the decision logic, and a refactoring comparison document.

Assignment 3: Iterative Algorithms, Pattern Generation & Series Approximation

Focus: Loops (`for`, `while`, `do-while`), Loop Control (`break`, `continue`), Nested Loops, Algorithmic Complexity

- **Objective:** Master loop control, invariant maintenance, nested iteration, and mathematical approximations using loop-based convergence.
- **Tasks:**
  1. **Mathematical Series & Convergence:**

- Implement programs to calculate mathematical constants (π, e) or functions ($\sin(x)$, $\cos(x)$ using Taylor Series expansions) up to a user-specified tolerance limit $\epsilon = 10^{-6}$.
- Use loop control strategies to stop iteration dynamically when consecutive terms differ by less than ϵ.

2. **Nested Loop Pattern & Matrix Engine:**
   - Write a program using nested loops to generate complex dynamic geometric patterns (e.g., Pascal's Triangle, Floyd's Triangle, or rotated Diamond patterns) based on user-defined height parameters.
   - Compute prime numbers up to N using nested loops (Trial Division) and compare performance against an optimized loop structure.
3. **Interactive Menu-Driven Loop Application:** Build a complete terminal-based interactive application (e.g., ATM Simulator, Inventory Tracker) that runs continuously inside a state-driven loop until an explicit exit command is triggered.

- **Deliverable:** Source code for series approximation, pattern generators, and an execution trace analyzing iteration efficiency.

Reference Books and Extended Learning

1. Wesley J. Chun, "***Core Python Applications Programming***", 3rd Edition , Pearson Education, 2016

2. Charles Dierbach, "***Introduction to Computer Science using Python***", Wiley, 2015

3. Jeeva Jose, "***Taming Python by Programming***", Khanna Publishers, New Delhi, 2018

4. Downey, A. et al., "***How to think like a Computer Scientist: Learning with Python***", John Wiley, 2015

Course Code	**26PLMCA106**		
Course Name	**DATA STRUCTURES LAB**		
Teaching Hours	L:0 T:0 P:45 SL: 15	**Course Credits**	2
Continuous Internal Assessment Marks	50	**Semester End Exam Marks**	50
Exam Hours	3 Hours	**Total Marks**	100
Course Type	PCL	**Course Category**	PCL
Prerequisites	Basic Data Structures, Knowledge of any programming language, preferably 'C'.		

Course Objectives

This is the companion course of 20MCA105 Advanced Data Structures and provides the students hands-on experience of the advanced data structures which will boost up the knowledge and confidence of students in applying these techniques while dealing with real life computing problems.

Course Outcomes (COs)

CO#	CO Statement	Knowledge Level
CO1	Use Debuggers, Profilers and advanced Compiler options.	K3
CO2	Implement the Set and Disjoint Set Data Structures.	K3

CO3	Understand the practical aspects of Advanced Tree Structures.	K2
CO4	Realize Modern Heap Structures for effectively solving advanced Computational problems.	K3
CO5	Implement Advanced Graph algorithms suitable for solving advanced computational problems.	K3

Course Articulation Matrix
CO-PO Mapping

CO#	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	2	-	-	-	3	–	1	–	–	–	–
CO2	3	2	2	-	1	–	-	–	–	–	–
CO3	2	2	3	2	1	–	1	–	–	–	–
CO4	2	3	3	2	1	–	1	–	–	–	–
CO5	3	3	2	1	2	–	1	–	–	–	–
Avg	**2.4**	**2.2**	**2.2**	**1.4**	**1.6**	**-**	**1**	**-**	**-**	**-**	**-**

1: Slight (Low), 2: Moderate (Medium), 3: Substantial (High), -: No Correlation,

Assessment Pattern

Bloom's Category	Continuous Assessment Tests		End Semester Examination
	Test 1	Test 2	
Remember (K1)			
Understand (K2)			
Apply (K3)	10	10	10
Analyse (K4)	10	10	10
Evaluate (K5)	10	10	10
Create (K6)	20	20	20

Mark distribution

Total Marks	Continuous Internal Evaluation (CIE)	End Semester Examination (ESE)	ESE Duration
100	50	50	3 Hours

Continuous Internal Evaluation (CIE) Pattern

Maximum Marks: 50	
Attendance	15%
Maintenance of daily lab record and GitHub management	20%
Regular class viva	15%
Timely completion of day to day tasks	20%
Tests/Evaluation	30%

End Semester Examination Pattern:

Maximum Marks: 50			
Verification of Daily program record and Git Repository			5 marks
Viva			10 marks
Problem solving (Based on difficulty level, one or more questions may be given)	Flowchart / Algorithm / Structured description of problem to explain how the problem can be solved / Interface Design	15%	35 marks
	Program correctness	50%	
	Code efficiency	15%	
	Formatted output and Pushing to remote Git	20%	

	repository		
Total Marks			50 marks

Syllabus:

Based on the syllabus of 26PCMCA103 Advanced Data Structures.

Suggested Assignments for Course Teacher's Assessment (CTA) Through Self-Learning:

Assignment 1: Binomial Heaps — Structural Mechanics, Amortized Analysis & Operations
Focus: Definition of Binomial Trees (B_k), Binomial Heap Invariants, Operations (Insertion, Union, Extract-Min, Decrease-Key, Delete), Structural Proofs

- **Objective:** Master the properties of Binomial Trees, trace pointer-level heap transformations, and analyze worst-case time bounds for multi-tree heap consolidation.
- **Tasks:**
  1. **Structural Analysis & Proofs:**
     - Prove by induction that a Binomial Tree B_k has exactly 2^k nodes, height k, and $\binom{k}{i}$ nodes at depth i.
     - Show why a Binomial Heap storing n elements consists of at most $\lfloor \log_2 n \rfloor + 1$ Binomial Trees, directly corresponding to the binary representation of n.
  2. **Step-by-Step Hand Tracing (Union & Consolidation):**
     - Construct two distinct Binomial Heaps:
       - H_1 containing keys [12, 7, 25, 15, 30, 8]
       - H_2 containing keys [10, 3, 18, 2, 45, 33, 14]
     - Trace the **UNION(H_1, H_2)** operation step-by-step. Show root-list merging and tree link operations (BINOMIAL-LINK) for equal-degree trees.
     - On the resulting merged heap, trace an **EXTRACT-MIN** operation. Draw the state of the heap after root removal, child reversals, and final consolidation.
  3. **Key Modifications:**
     - Trace a **DECREASE-KEY** operation that triggers rootward bubbles, followed by a **DELETE** operation executed via DECREASE-KEY to $-\infty$ followed by EXTRACT-MIN.
- **Deliverable:** A technical write-up detailing the mathematical proofs, clear step-by-step execution diagrams for UNION and EXTRACT-MIN, and worst-case complexity derivations.

Assignment 2: Fibonacci Heaps — Lazy Consolidations & Amortized Complexity
Focus: Fibonacci Heap Structural Properties, Cascading Cuts, Lazy Operations, Potential Method for Amortized Analysis

- **Objective:** Understand how lazy updates achieve $O(1)$ amortized bounds for INSERT, UNION, and DECREASE-KEY, and evaluate how cascading cuts maintain maximum degree bounds.
- **Tasks:**

1. **Theoretical Amortized Analysis (The Potential Method):**
    - Define the potential function $\Phi(H) = t(H) + 2c(H)$, where $t(H)$ is the number of trees in the root list and $c(H)$ is the number of marked nodes.
    - Derive the amortized time complexity for:
        - **INSERT** ($O(1)$ amortized)
        - **DECREASE-KEY** ($O(1)$ amortized, proving why marked nodes and cascading cuts prevent tree heights from becoming degenerate)
        - **EXTRACT-MIN** ($O(\log n)$ amortized)
2. **Execution Trace of Cascading Cuts:**
    - Starting with a fully consolidated Fibonacci Heap, trace a series of **DECREASE-KEY** operations that trigger single and multi-level **cascading cuts**. Draw the state of the root list, child lists, and node marks (marked vs. unmarked) after every step.
3. **Structural Comparison:** Prove why $O(1)$ amortized DECREASE-KEY in Fibonacci Heaps yields theoretical asymptotic speedups for graph algorithms like Dijkstra's Single Source Shortest Path ($O(V \log V + E)$ vs $O(E \log V)$) and Prim's Minimum Spanning Tree.

- **Deliverable:** Mathematical derivations of amortized time bounds using potential functions, step-by-step diagrams of cascading cuts, and a formal runtime analysis comparing Dijkstra with standard vs. Fibonacci Heaps.

Text Books

1. Cormen T.H., Leiserson C.E, Rivest R.L. and Stein C, ***Introduction to Algorithms,*** Prentice Hall India, New Delhi, 2004

Reference Books and Extended Learning

1. Kleinberg, Jon, and Eva Tardos. ***Algorithm design***. Pearson Education India, 2006.
2. Aho A.V., Hopcroft J.E., and Ullman J.D., ***Data Structures and Algorithms***, Pearson Education, New Delhi, 1983.
3. Sahni S., Data Structures, ***Algorithms, and Applications in C++***, Mc Graw Hill, Singapore, 1998.

Course Code	26SEMCA107		
Course Name	**ENTREPRENEURSHIP & INNOVATIONS IN TECHNOLOGY**		
Teaching Hours	L:15 T:0 P:0 SL:15	**Course Credits**	1
Continuous Internal Assessment Marks	100	**Semester End Exam Marks**	Nil
Exam Hours	Nil (100% Internal)	**Total Marks**	100
Course Type	CC	**Course Category**	SEC
Prerequisites	Nil		

Course Objectives

- To introduce students to the fundamentals of entrepreneurship, innovation and the technology-startup ecosystem.
- To develop an entrepreneurial mindset and design-thinking skills to identify problems and ideate technology-based solutions.
- To familiarize students with business-model design, funding avenues, and the IPR/legal aspects relevant to launching a technology venture.

Course Outcomes (COs)

CO#	CO Statement	Knowledge Level

CO1	Understand the concepts, process and significance of entrepreneurship and innovation in the technology sector.	K2
CO2	Apply design-thinking techniques to identify real-world problems and ideate innovative technology-based solutions.	K3
CO3	Analyse the startup ecosystem, business models and funding options available for a technology venture.	K4
CO4	Develop a basic business plan / pitch deck for a technology product or service idea.	K6
CO5	Demonstrate awareness of intellectual property rights (IPR) and the legal/ethical aspects relevant to startups.	K2

Course Articulation Matrix

CO-PO Mapping

CO#	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO 9	PO10	PO11
CO1	2	1	1	-	-	-	2	1	-	-	-
CO2	2	3	2	1	-	1	2	1	-	-	-
CO3	3	2	3	1	-	1	2	2	-	-	-
CO4	2	3	3	2	1	2	2	2	-	-	-
CO5	1	1	2	1	-	-	3	2	-	-	-
Avg	**2.0**	**2.0**	**2.2**	**1.3**	**1.0**	**1.3**	**2.2**	**1.6**	**-**	**-**	**-**

1: Slight (Low), 2: Moderate (Medium), 3: Substantial (High), -: No Correlation,

Assessment Pattern

Bloom's Category	Series Test 1 (Modules 1-3)	Series Test 2 (Modules 4-5)
Remember (K1)	6	4
Understand (K2)	8	6
Apply (K3)	8	8
Analyse (K4)	4	6
Evaluate (K5)	2	4
Create (K6)	2	2

Total	**30**	**30**

Mark distribution

Total Marks	Continuous Internal Evaluation (CIE)	Semester End Examination (SEE)	Exam Duration
100	100	Nil	Nil

Continuous Internal Evaluation (CIE) Pattern

Series Test 1 - Theory : 30
Assessment 1 - Activity / Seminar / Assignment : 20
Series Test 2 - Theory : 30
Assessment 2 - Activity / Seminar / Assignment : 20
Total : 100

Course Syllabus

Total Hours: 30 (15 hours core syllabus + 15 hours Activity / Seminar / Innovative-project component, distributed module-wise below.)

Module: 1	**Contact Hours: 3**
Introduction to Entrepreneurship: Concept, need and characteristics of an entrepreneur; Entrepreneurship vs. Intrapreneurship; Types of entrepreneurship (technology, social, women, rural); Role of entrepreneurship in economic development; Technology entrepreneurship and its relevance in the digital economy.	
Activity / Seminar / Innovative Component	**Hours: 3**
• Seminar presentation on a successful entrepreneur / technology startup, highlighting the entrepreneurial journey. • Entrepreneurial-mindset self-assessment exercise followed by a reflective write-up.	
Module: 2	**Contact Hours: 3**
Innovation & Design Thinking: Innovation - meaning and types (incremental vs. disruptive); Design-thinking process - Empathize, Define, Ideate, Prototype, Test; Idea-generation techniques (brainstorming, SCAMPER, mind-mapping).	
Activity / Seminar / Innovative Component	**Hours: 3**
• **Pitch Icebreaker** – Each student will spontaneously pitch, in under a minute, a small real-life problem they have personally faced along with a possible solution, in order to build comfort with opportunity-spotting and quick articulation of ideas. • Group discussion / debate on 'Incremental vs. Disruptive Innovation' using real examples.	

Module: 3	**Contact Hours: 3**
Startup Ecosystem & Business Models: Lean Startup approach and Minimum Viable Product (MVP); Business Model Canvas and revenue models; Startup ecosystem in India - incubators, accelerators, Startup India scheme, MSME support schemes; Role of college/university Innovation & Incubation Cells.	
Activity / Seminar / Innovative Component	**Hours: 3**
• Group activity: prepare a Business Model Canvas for a technology idea ideated in Module 2. • Field study / guest talk / virtual visit report on a local incubator, accelerator, or startup.	
Module: 4	**Contact Hours: 3**
Funding & Pitching: Sources of funding - bootstrapping, angel investors, venture capital, crowdfunding, government schemes; Basic financial planning for a startup; Fundamentals of pitching to investors.	
Activity / Seminar / Innovative Component	**Hours: 3**
• Mock investor-pitch / innovative-project demo day - each team presents a 3-5-minute pitch deck before a panel of faculty/peers for feedback. • Assignment: identify and compare suitable funding options for a chosen technology idea/Startup Failure Analysis and Financial Planning for a Technology Venture.	

Module: 5	**Contact Hours: 3**
IPR, Legal & Ethical Aspects: Basics of Intellectual Property Rights - patents, copyrights, trademarks, trade secrets, relevant to technology products; Legal and ethical considerations in starting up; Overview of business registration and compliance (GST, MSME/Udyam registration).	
Activity / Seminar / Innovative Component	**Hours: 3**
• Case-Study cum Peer Webinar on IPR – Students will choose a technology product, identify the patent, trademark, and copyright protections applicable to it, and present their findings as a case study or webinar to their peers. • Quiz / class discussion on ethical dilemmas faced by technology startups.	

Suggested Seminar / Activity / Innovative Project Topics (Overall Pool)

- Startup Case Study Seminar: Research and present the founding story, business model and innovation strategy of a technology startup of choice.
- Design-Thinking Mini-Hackathon: Form teams to ideate and prototype a solution for a campus or community problem.
- Business Model Canvas Workshop: Develop and present a Business Model Canvas for an original technology product/service idea.

- Mock Investor Pitch: Prepare and deliver a 3-5-minute pitch deck (problem, solution, market, business model, ask) before a panel.
- IPR Case Study: Analyse a real patent/trademark dispute involving a technology company and present the key learnings.
- Incubator/Accelerator Visit Report: Visit (physically or virtually) a startup incubator/accelerator and submit a reflective report.
- Innovative Project Demo: Build and demonstrate a small working prototype or proof-of-concept addressing a real-world problem.
- Reflective Journal: Maintain a short journal reflecting on entrepreneurial-mindset exercises undertaken through the course.

Text Books

1. Rajeev Roy, Entrepreneurship, 2nd Edition, Oxford University Press, 2011.

2. Poornima M. Charantimath, Entrepreneurship Development and Small Business Enterprises, 2nd Edition, Pearson Education, 2018.

3. Bruce R. Barringer and R. Duane Ireland, Entrepreneurship: Successfully Launching New Ventures, 6th Edition, Pearson, 2018.

Reference Books and Extended Learning

1. Eric Ries, The Lean Startup, Crown Business, 2011.

2. Alexander Osterwalder and Yves Pigneur, Business Model Generation, Wiley, 2010.

3. Steve Blank and Bob Dorf, The Startup Owner's Manual, K&S Ranch, 2012.

4. Vasant Desai, Dynamics of Entrepreneurial Development and Management, Himalaya Publishing House.

5. Web Resources: Startup India Portal (www.startupindia.gov.in); MSME/Udyam Registration Portal (udyamregistration.gov.in); World Intellectual Property Organization (www.wipo.int).