COURSE CURRICULUM

for

B.TECH. DEGREE

in

INFORMATION TECHNOLOGY

(Applicablefromtheacademicsession2024-2025)

DR. B.C. ROY SOCIETY ENGINEERING AN AUTONOMOUS INSTITUTE

Dr.B.C.RoyEngineeringCollege

An Autonomous Institution

Approvedby:AllIndiaCouncilforTechnicalEducation(AICTE)

Affiliated to: MaulanaAbulKalam Azad University of Technology, West Bengal(FormerlyKnown as -WBUT)

Jemua Road, Durgapur, West Bengal, India,713206

Course Name: Computer organization and architecture
Course Code: PCC-IT-401
(Semester- IV)
Course Broad Category: Information Technology

……………………………………………………………………………………………………
…

1. **Course Prerequisite:**
   Number system concept, Boolean Algebra, Basic components of a digital computer and fundamentals of programming Binary Arithmetic as covered in Basic Computation & Principles of Computer Programming, Digital electronics.

2. **Course Learning Objectives:**
   Explain the structural and functional organization of a computer system. Define the addressing modes, instruction formats. Demonstrate different circuit designs using basic gates and hardware architectures. Discuss the integer and floating-point number representations and the operations applied on it. Analyze various components of memory hierarchy and I/O data transfer.

3. **Teaching methodology and evaluation system for the course:**
   Teaching methodology – Lectures and Presentations, Interactive Discussions and Case Studies.

   Evaluation System –

A. Mid-Term Exam (20 Marks)- Summative Assessment (CIA-1)
B. Internal Assessment (20 Marks)- Formative Continuous Assessment [Continuous Assessment 1 (CIA-2)]
C. End-Semester Exam (60 Marks)- Summative Assessment.

4. **Course Content:**
   Course Name: Computer organization and architecture
   Course Code: PCC-IT-401

Hours per Week: 3L: 0T: 0P
Credits: 3

Module number	Topics	Lecture number
1	Evolution of Computer Systems Number Representation, Commonly used number systems. Fixed and floating point representation of numbers. Floating point arithmetic Basic organization of the stored program computer and operation sequence for execution of a program. Basic Operation of a Computer, Concept of operator, operand, registers and storage. Instruction cycle: Fetch, decode and execute cycle, Instruction format. Instruction sets and addressing modes. Software and Architecture Types, Role of operating systems and compiler/assembler. Floating point - IEEE 754 standard. Floating point numbers, floating point arithmetic. Fixed point multiplication –Booth's algorithm.	10
2	Fixed point division - Restoring and non-restoring algorithms. Design of adder, Design of multiplier. Instruction Format and Addressing Modes Instruction Set Architecture, MIPS32 Instruction Set, MIPS Programming Examples Memory Addressing and Languages. Von Neumann architecture Quantitative techniques in computer design, measuring and reporting performance. Amdahl's law. CISC and RISC Architecture.	9
3	Pipelining: Basic concepts, Types of pipeline, instruction and arithmetic pipeline. Data hazards, control hazards and structural hazards, techniques for handling hazards. Exception handling. Pipeline optimization techniques; Compiler techniques for improving performance. Instruction-level parallelism: basic concepts, techniques for increasing ILP, VLIW processor Superscalar, super pipelined architectures. Array and vector processors.	6
4	Memory unit design with special emphasis on implementation of CPU-memory interfacing. Memory organization, memory hierarchy. Static and dynamic memory, Data path design for read/write access. Cache memory organization, Hierarchical memory technology: Inclusion, Coherence and locality properties; Techniques for reducing cache misses; Virtual memory organization, Memory mapping, direct mapping, Associative memory, set associative mapping Memory replacement policies.	7
5	Secondary storage devices Input output organization, Data transfer technique, Interrupt handling techniques, Design of control unit - hardwired and microprogrammed control. I/O operations - Concept of handshaking, Polled I/O, DMA. Multiprocessor architecture: taxonomy of parallel architectures; Centralized shared- memory	8

5. References:

Text Book:

- M. Morris Mano, Rajib Mal—Computer system architecture; **Publisher. Pearson Education India.**
- Kai Hwang --- Advanced Computer architecture; **Publisher.** McGraw Hill Education**.**

Reference Books:

1. V. Carl, G. Zvonko and S. G. Zaky, “Computer organization”, McGraw Hill, 1978.
2. B. Brey and C. R. Sarma, “The Intel microprocessors”, Pearson Education, 2000.
3. J. L. Hennessy and D. A. Patterson, “Computer Architecture A Quantitative Approach”, Morgan Kauffman, 2011.
4. W. Stallings, “Computer organization”, PHI, 1987.
5. P. Barry and P. Crowley, “Modern Embedded Computing”, Morgan Kaufmann, 2012.
6. N. Mathivanan, “Microprocessors, PC Hardware and Interfacing”, Prentice Hall, 2004.
7. Y. C. Lieu and G. A. Gibson, “Microcomputer Systems: The 8086/8088 Family”, Prentice Hall India, 1986.
8. J. Uffenbeck, “The 8086/8088 Design, Programming, Interfacing”, Prentice Hall, 1987.
9. B. Govindarajalu, “IBM PC and Clones”, Tata McGraw Hill, 1991.
10. P. Able, “8086 Assembly Language Programming”, Prentice Hall India6. Winfried Karl Grassmann and Jean-Paul Tremblay, Logic and Discrete Mathematics, PEARSON.

6. Course Outcomes (CO):

Course Outcomes	**Details/Statement**	**Action Verb**	**Knowledge Level**
PCC-IT 401.1	Demonstrate different concepts of computer architecture to improve the performance of a computer. Discuss the integer and floating-point number representations and the operations applied on it.	Identify	Remember
PCC-IT 401.2	Analyze various components of memory hierarchy. Cache mapping and idea about data movement.	Explain	Understand
PCC-IT 401.3	Apply different methods for proper organization of memory in computer architecture.	Implement	Apply

PCC-IT 401.4	Interpret an architectural problem to use accurate method to solve it.	Organize	Analyze
PCC-IT 401.5	Explain the concept of I/O interfacing and various taxonomy of I/O data transfer.	Assess	Evaluate
PCC-IT 401.6	Evaluate performance of a pipeline, data hazard, and memory performance.	Construct	Create

7. Mapping of course outcomes to module / course content

Module	**CO1**	**CO2**	**CO3**	**CO4**	**CO5**	**CO6**
1	3	-	-	2	-	1
2	2	3	-	1	-	1
3	2	3	3	2	-	1
4	3	-	-	2	-	1
5	2	-	-	1	3	1

8. Mapping of the Course outcomes to Program Outcomes (PO)

	PO1	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO1 0**	**PO1 1**	**PO1 2**
CO1	2	2	1	1	2	-	-	-	-	-	-	1
CO2	1	2	3	1	-	-	-	-	-	-	-	1
CO3	1	2	2	1	1	-	-	-	-	-	-	1
CO4	1	2	1	1	2	-	-	-	-	-	-	2
CO5	2	2	2	2	3	-	-	-	-	-	-	1
CO6	1	1	1	1	-	-	-	-	-	-	-	1

9. Mapping to Program Specific Outcome (PSO)

	PSO1	**PSO2**	**PSO3**	**PSO4**
CO1	3	1	2	-
CO2	3	1	2	-
CO3	3	-	1	-
CO4	2	1	3	-
CO5	2	-	2	1
CO6	1	-	2	3

***** End of Syllabus*****

Course Name: Discrete Mathematics
Course Code: PCC-IT-402
(Semester IV)
Course Broad Category: B. Tech IT

…………………………………………………………………………………………………

1. **Course Prerequisite:**
   Concept of Mathematics in the previous semesters of B. Tech curriculum.

2. **Course Learning Objectives:**
   Introduces the elementary discrete mathematics for computer science and engineering. Topics include formal logic notation, methods of proof, induction, sets, relations, graph theory, permutations and combinations, counting principles; recurrence relations and generating functions.

3. **Teaching methodology and evaluation system for the course:**
   Teaching methodology – Lectures and Presentations, Interactive Discussions and Case Studies, Guest Lectures.

   Evaluation System –

A. Mid-Term Exam (20 Marks)- Summative Assessment (CIA-1)
B. Internal Assessment (20 Marks)- Formative Continuous Assessment [Continuous Assessment 1 (CIA-2)]
C. End-Semester Exam (60 Marks)- Summative Assessment.

4. **Course Content:**
   Course Name: Discrete Mathematics
   Course Code: BPCC-IT-402
   Hours per Week: 3L: 0T: 0P
   Credits: 3

Module	Topics	45L
1.	**Mathematical Logic and Relations**: Statements and notations, Connectives, Well-formed formulas, Truth Tables, tautology, equivalence implication, Normal forms, Quantifiers, universal quantifiers. Properties of binary relations, equivalence relations, partial ordering relations, Lattices, Hasse diagram. **Predicates**: Predicative logic, Free & Bound variables, Rules of inference, Consistency, proof of contradiction, Automatic Theorem Proving.	6L
2.	**Algebraic structures:** Algebraic systems, examples and general properties, Semigroups, Monoids and Groups, different Finite and Infinite groups. **Elementary Combinatorics:** Basis of counting, Sum rule and Product rule, Combinations & Permutations, Binomial Coefficients, Binomial Multinomial theorems, the principles of Inclusion – Exclusion. Pigeon hole principles and its application.	10L

Module	Topics	45L
3.	**Recurrence Relation:** Generating Functions, Calculating Coefficient of generating function, Recurrence relations, solving recurrence relation by substitution and Generating functions, Characteristics roots solution of In-homogeneous Recurrence Relation.	5L
4.	**Basic Graph Theory:** **Basic Concept of graph theory**- Definitions and Matrix Representation: Incidence & Adjacency matrix. Weighted graph, Connected and disconnected graphs, complement of a graph, Regular graph, Complete graph, Sub-graph, Walk, Path, Circuit, Euler and Hamiltonian graph- Necessary conditions and sufficient conditions, di-graph, cut sets and cut vertices, Graph isomorphism, Bipartite Graphs- properties of Bipartite Graphs, Dijkstra's Algorithm for shortest path problem. **Graph Operations:** Union, Sum, Cartesian Product, Composition, Graphic sequences, Havel- Hakimi criterion, Realization of a graphic sequence.	10L
5.	**Advanced Graph Theory:** **Tree:** Definitions and characterizations, Number of trees, Cayley's formula, Kirchoff-matrix-tree theorem, **Spanning tree:** Definition, Number of minimum spanning trees, BFS and DFS algorithms, Minimal spanning tree- Kruskal's and Prim's algorithms. Eulerian Graphs, Fleury's algorithm, Chinese Postman problem, Hamilton Graphs, Introduction, Necessary conditions and sufficient conditions. **Planar graphs-** Definitions and theorems, Kuratowski's theorem for testing planarity, Duality in planar graphs. **Graph coloring:** Vertex Colorings- Basic definitions, Cliques and chromatic number, Greedy coloring algorithm, Brooks theorem, Edge Colorings. **Independent sets coverings and matchings: -** Introduction, Independent sets and coverings: basic equations, Matchings in bipartite graphs, Hall's Marriage Theorem (Statement only), K¨onig's Theorem (Statement only), Perfect matchings in graphs, Greedy and approximation algorithms.	14L

5. References:

Text Book:

- Rosen, Kenneth H.: Discrete Mathematics and its Applications with Combinatorics and Graph Theory (7th Edition), TMH (Tata McGraw-Hill).
- Mott, Joe L., Kandel, Abraham, and Baker, Teodore P.: Discrete Mathematics for Computer Scientists and Mathematicians, Pearson Education.
- Johnsonbaugh, Richard: Discrete Mathematics, Pearson Education.
- Chandrasekaran, N., and Umaparvathi, M.: Discrete Mathematics, PHI Learning.
- J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications*, Macmillan Press.

Reference Books:

- Tremblay, J.P., and Manohar, R.: Discrete Mathematical Structures with Applications to Computer Science, TMH (Tata McGraw-Hill).
- Mott, Joe L., Kandel, Abraham, and Baker, Teodore P.: Discrete Mathematics for Computer Scientists and Mathematicians (2nd Edition), Pearson Education.
- Johnsonbaugh, Richard: Discrete Mathematics (7th Edition), Pearson Education.
- Goodaire, Edgar G., and Parmenter, Michael M.: Discrete Mathematics with Graph Theory, Pearson Education.

- Grimaldi, Ralph P.: Discrete and Combinatorial Mathematics: An Applied Introduction (5th Edition), Pearson Education.
- D.B. West, Introduction to Graph Theory, Pearson Education.
- K.R. Parthasarathy, Basic Graph Theory, Tata McGraw Hill Education.
- Sudarsan Nanda, Graph Theory and Algorithms, Allied Publishers.
- M. Mukundarajan, Graph Theory with Applications to Computer Science and Engineering, Yes Dee Publishing.

6. Course Outcomes (CO):

Course Outcomes	Details/Statement	Action Verb	Knowledge Level
PCC-IT-402.1	Ability to understand and construct precise mathematical proofs.	Identify	Remember
PCC-IT-402.2	Ability to use logic and set theory to formulate precise statements.	Explain	Understand
PCC-IT-402.3	Ability to analyze and solve counting problems on finite and discrete structures	Implement	Apply
PCC-IT-402.4	Ability to apply graph theory in solving problems in computer science and information technology.	Organize	Analyze
PCC-IT-402.5	Ability to apply the concept of advanced graph theory in networking and shortest path problems.	Assess	Evaluate
PCC-IT-402.6	Build up logical and analytical skills to create a new idea appreciated by academics, research & emerging trends in industry.	Construct	Create

7. Mapping of course outcomes to module / course content

Module	CO1	CO2	CO3	CO4	CO5	CO6
1	3	2	1	-	-	1
2	3	3	2	-	-	1
3	1	1	3	-	-	1
4	2	1	2	3	-	1
5	2	1	1	3	3	1

8. Mapping of the Course outcomes to Program Outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO 11	PO 12
CO1	2	2	1	1	2	-	-	-	-	-	-	1
CO2	1	2	3	1	-	-	-	-	-	-	-	1
CO3	1	2	2	1	1	-	-	-	-	-	-	1
CO4	1	2	1	1	2	-	-	-	-	-	-	2
CO5	2	2	2	2	3	-	-	-	-	-	-	1
CO6	1	1	1	1	-	-	-	-	-	-	-	1

9. Mapping to Program Specific Outcome (PSO)

	PSO1	PSO2	PSO3	PSO4
CO1				
CO2				
CO3				
CO4				
CO5				
CO6				

Course Name: Design and Analysis of Algorithms
Course Code: PCC- IT-403
(Semester- IV)
Course Broad Category: PCC-MAJOR

…………………………………………………………………………………………………

1. **Course Prerequisite:**
   Concept of Data Structure and algorithms, and basic programming ability

2. **Course Learning Objectives:**

Upon completion of this course, students will be able to do the following:

- Analyze the asymptotic performance of algorithms.
- Write rigorous correctness proofs for algorithms.
- Demonstrate a familiarity with major algorithms and data structures.
- Apply important algorithmic design paradigms and methods of analysis.
- Synthesize efficient algorithms in common engineering design situations.

3. **Teaching methodology and evaluation system for the course:**

Teaching methodology –Lectures and Presentations, Interactive Discussions and Case Studies, Guest Lectures.
Evaluation System –

A. Mid-Term Exam (20 Marks)- Summative Assessment (CIA-1)
B. Internal Assessment (20 Marks)- Formative Continuous Assessment [Continuous Assessment 1 (CIA-2)]
C. End-Semester Exam (60 Marks)- Summative Assessment.

4. **Course Content:**

Course Name: Design and Analysis of Algorithms
Course Code: PCC-IT-403
Hours per Week: 3L:0T:0P
Credits: 3

Module	Topics	45L
1.	Introduction: Characteristics of algorithm. Analysis of algorithm: Asymptotic analysis of complexity bounds – best, average and worst-case behavior; Performance measurements of Algorithm, Time and space trade-	9L

Module	Topics	45L
	offs, Analysis of recursive algorithms through recurrence relations: Substitution method, Recursion tree method and Masters' theorem.	
2.	Fundamental Algorithmic Strategies: Brute-Force, Greedy, Dynamic Programming, Branch and- Bound and Backtracking methodologies for the design of algorithms; Illustrations of these techniques for Problem-Solving, Bin Packing, Knap Sack TSP. Heuristics –characteristics and their application domains.	10L
3.	Graph and Tree Algorithms: Traversal algorithms: Depth First Search (DFS) and Breadth First Search (BFS); Shortest path algorithms, Transitive closure, Minimum Spanning Tree, Topological sorting, Network Flow Algorithm	10L
4.	Tractable and Intractable Problems: Computability of Algorithms, Computability classes –P,NP, NP- complete and NP-hard. Cook's theorem, Standard NP-complete problems and Reduction techniques.	8L
5.	Advanced Topics: Approximation algorithms, Randomized algorithms, Class of problems beyond NP – P SPACE	8L

5. Text Book:

- Introduction to Algorithms, 4TH Edition, Thomas H Cormen, Charles E Lieserson, Ronald L Rivest and Clifford Stein, MIT Press/McGraw-Hill.
- Fundamentals of Algorithms – E. Horowitz et al.
- Algorithm Design, 1ST Edition, Jon Kleinberg and ÉvaTardos, Pearson.
- Algorithm Design: Foundations, Analysis, and Internet Examples, Second Edition, Michael T Goodrich and Roberto Tamassia, Wiley.

Reference Books:

- Fundamentals of Computer Algorithms by Sartaj Sahni and Sanguthevar Rajasekaran Ellis Horowitz, University Press.
- Design & Analysis of Algorithms
  by Biswajit Bhowmik, Katson Publication.

6. Course Outcomes (CO):

On completion of the course students will be able to

Course Outcomes	Details/Statement	Action Verb	Knowledge Level
PCC-IT404.CO1	Analyze the complexities of different algorithms.	Analyze	K4
PCC-IT404.CO2	Develop the algorithm techniques (example Divide &amp; Conquer, Dynamic Programming etc) to solve different Mathematical models.	Develop	K3
PCC-IT404.CO3	Illustrate the techniques of Greedy paradigm, Branch and Bound, Backtracking etc and compare and contrast them.	Illustrate	K2
PCC-IT404.CO4	Discuss the types of Minimal spanning tree and traversal algorithm with their applications.	Discuss	K6
PCC-IT404.CO5	intractable problems to introduce polynomial and non polynomial reduction.	Understand	K2
PCC-IT404.CO6	Explain the randomized algorithms and approximation algorithms to illustrate their applications.	Construct	K5

7. Mapping of course outcomes to module / course content

Module	CO1	CO2	CO3	CO4	CO5	CO6
1						
2						
3						
4						
5						
6						

8. Mapping of the Course outcomes to Program Outcomes (PO)

	PO1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO1 0	PO1 1	PO1 2
CO1												
CO2												
CO3												
CO4												
CO5												
CO6												

9. Mapping to Program Specific Outcome (PSO)

	PSO1	PSO2	PSO3	PSO4
CO1				
CO2				
CO3				
CO4				
CO5				
CO6				

***** End of Syllabus*****

Course Name: Formal Language & Automata Theory
Course Code: PCC-IT-404
(Semester– IV)
Course Broad Category: Engineering Science

……

1. Course Prerequisite:

Elementary discrete mathematics including the notion of set, function, relation, product, partial order, equivalence relation, graph & tree.

2. Course Learning Objectives:

I. To introduce the classification of machines by their power to recognize languages and to solve problems in computing with compiler design.
II. To familiarize how to employ deterministic and non-deterministic machines.
III. Identify different methods of lexical analysis and design top-down and bottom-up parsers.
IV. Develop algorithms to generate code for a target machine.

3. Teaching methodology and evaluation system for the course:

Teaching Methodology –Lectures and Presentations, Interactive Discussions and Case Studies.

Evaluation System –

A. Mid-Term Exam (20 Marks)- Summative Assessment (CIA-1)
B. Internal Assessment (20 Marks)- Formative Continuous Assessment [Continuous Assessment 1 (CIA-2)]
C. End-Semester Exam (60 Marks)- Summative Assessment.

4. Course Content:

Course Name: Formal Language & Automata Theory
Course Code: PCC-IT-404
Hours per Week: 3L: 0T: 0P
Credits: 3

Module	Topics	Lectures
1	**Fundamentals:** Definition of Automata, Use of Automata. Definition of sequential circuit, block diagram, mathematical representation, and concept of transition table and transition diagram (Relating of Automata concept to sequential circuit concept) Design of sequence detector, Introduction to finite state model **Finite state machine with Compiler:** Definitions, capability & state equivalent, Finite memory definiteness, testing table & testing graph. Limitations of FSM Application of finite automata, Finite	5L

	Automata with Output-Moore & Mealy machine. Compilers, Analysis of the source program, The phases of the compiler, Cousins of the compiler.	
2	**Deterministic finite automaton and Non-deterministic finite automaton:** Transition diagrams and Language recognizers. Chomsky Hierarchy. Finite Automata: NFA with e transitions - Significance, acceptance of languages. NFA to DFA conversion. DFA minimization theorem.	4L
3	**Regular Languages:** Regular sets. Regular expressions, identity rules. Arden _s theorem state and prove Constructing Finite Automata for a given regular expression, Regular string accepted by NFA/DFA. Pumping lemma of regular sets. **Grammar Formalism:** Regular grammars-right linear and left linear grammars. Equivalence between regular linear grammar and FA.	4L
4	**Introduction to Context free grammars:** Derivation trees, sentential forms. Right most and leftmost derivation of strings. Basic applications of the concept of CFG, Ambiguity in context free grammars. **Push down Automata:** Push down automata, definition. Acceptance of CFL, Acceptance by final state and acceptance by empty state and its equivalence. Equivalence of CFL and PDA.	5L
5	**Turing Machine:** Turing Machine, definition, model, Design of TM, TM as language accepter, TM as transducers.	2L
6	**Lexical Analysis & Syntax Analysis:** The role of the lexical analyzer, Tokens, Patterns, Lexemes, Input buffering, Specifications of a token, Recognition of a tokens, The role of a parser, Context free grammars, Writing a grammar, Top down Parsing, Non- recursive Predictive parsing (LL), Error Recovery strategies for different parsing techniques.	6L
7	**Type checking:** Type systems, Specification of a simple type checker, Equivalence of type expressions, Type conversions, Source language issues, Storage organization. **Code optimization:** Introduction, Basic blocks & flow graphs, Transformation of basic blocks. **Code generations:** Issues in the design of code generator. Register allocation & assignment.	6L

5. References:

Textbooks:

a. "Theory of Computer Science-Automata Languages and Computation‖, Mishra andChandrashekaran,2nd edition, PHI".

b. "Switching & Finite Automata‖, ZVI Kohavi,2ndEdn.,TataMcGrawHill"

c. "An Introduction to Computing‖, Peter Linz, Narosa.

d. Aho, Sethi, Ullman - "Compiler Principles, Techniques and Tools" - Pearson Education.

e. Holub - "Compiler Design in C" - PHI.

Reference books:

a. "Introduction to Automata Theory Language and Computation‖, Hopcroft H.E.and Ullman J.D".
b. "Compilers: Principles, Techniques, and Tools", Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman.

6. Course Outcomes (CO):

After the completion of the course the student will be able to learn Automata and Compiler design concepts:

Course Outcomes	Details	Action Verb	Knowledge Level
PCC-IT-404.CO1	Understand situations in related areas of theory in computer science.	Understand	L-2
PCC-IT-404.CO2	Model, compare and analyze different computational models using combinatorial methods and Identify limitations of some computational models and possible methods of proving them.	Apply	L-3
PCC-IT-404.CO3	Analyze rigorously formal mathematical methods to prove properties of languages, grammars and Automata with specification design top-down and bottom-up parsers.	Apply	L-3
PCC-IT-404.CO4	Construct algorithms for different problems and argue formally about correctness on different restricted Machine models of computation.	Analyze	L-4
PCC-IT-404.CO5	Use algorithm to simplify grammar and understand given grammar specification develop the lexical analyser.	Create	L-5
PCC-IT-404.CO6	Design Turing Machine for the phrase-structured languages using algorithms to generate code for a target machine.	Understand	L-6

7. Mapping of course outcomes to module / course content

Module	CO1	CO2	CO3	CO4	CO5	CO6
1	3	-	-	-	-	-
2	-	3	-	-	-	-
3	-	-	3	-	-	-
4	-	-	-	3	-	-
5	-	-	-	-	3	-
6	-	-	-	-	5	-
7	-	-	-	-	-	6

8. Mapping of the Course outcomes to Program Outcomes (PO)

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	2										2

CO2	3	2										1
CO3	3	2										2
CO4	3	2										2
CO5	3	2										2
CO6	3	2										2

9. Mapping to Program Specific Outcome (PSO)

	PSO1	PSO2	PSO3	PSO4
CO1	3	2	2	-
CO2	3	3	2	-
CO3	3	3	3	-
CO4	3	3	3	-
CO5	3	3	3	2
CO6	3	3	3	2

***** End of Syllabus*****

Course Name: Object Oriented Programming
Course Code: PCC-IT- 405
(Semester- IV)
Course Broad Category: PCC- Major

....

1. **Course Prerequisite:**

   A) Basic Computer Skills: Familiarity with using a computer, file management, and navigating the operating system.
   B) Understanding of Programming Concepts: While not mandatory, having knowledge of basic programming concepts can be beneficial. This includes:
   - **Variables**: Understanding data storage and types (integers, strings, booleans, etc.).
   - **Control Structures**: Familiarity with conditionals (if statements) and loops (for, while).
   - **Functions/Methods**: Knowing how to define and call functions.

   C) Problem-Solving Skills: The ability to break down problems into smaller, manageable parts is crucial in programming.
   D) Familiarity with Development Tools: Knowing how to use an Integrated Development Environment (IDE) like Eclipse or IntelliJ IDEA can enhance the learning experience, but it can also be learned alongside Java.
   E) Basic Understanding of Data Structures: Familiarity with arrays, lists, and maps will aid in grasping Java's collection framework.

2. **Course Learning Objectives:**
   Adaptability - means the software may run on different generations and grows over a long lifetime. To reuse the code in which child classes uses the properties of the base class. To hold entire data into a single object such that it would be easy to pass this to other methods and objects. Multiprocessing is also in object-oriented programming. Binding of data to avoid data loss. Robustness - it means complex programs should operate correctly.

3. **Teaching methodology and evaluation system for the course:**

Teaching methodology –Lectures and Presentations, Interactive Discussions and Case Studies, Guest Lectures.

Evaluation System –

A. Mid-Term Exam (20 Marks)- Summative Assessment (CIA-1)
B. Internal Assessment (20 Marks)- Formative Continuous Assessment [Continuous Assessment 1 (CIA-2)]
C. End-Semester Exam (60 Marks)- Summative Assessment.

4. **Course Content:**
Course Name: Object Oriented Programming
Course Code: PCC-IT- 405
Hours per Week: 3L : 0T : 0P
Credits: 3

Module	Details	Contact Hours
I	**Object Oriented design** Concepts of object oriented programming language, Major and minorelements,Object,Class,relationshipsamongobjects,aggregation, links, relationships among classes-association, aggregation, using, instantiation, meta-class, grouping constructs.	2L
II	**Object Oriented concepts** Difference between OOP and other conventional programming– advantages and disadvantages. Class, object, message passing, inheritance, encapsulation, polymorphism	2L
III	**Basic concepts of Object Oriented Programming using JAVA** **Class &Object properties** Basic concepts of java programming–advantages of java, byte-code &JVM, data types ,access specifies, operators, control statements &loops, array, creation of class, object, constructor, finalize and garbage collection, use of method overloading, this keyword, use of objects as parameter &methods returning objects call by value &call by reference, static variables& methods, garbage collection, nested& inner classes, basic string handling concepts- concept of mutable and immutable string, command line arguments, basics of I/O operations –keyboard input using Buffered Reader &Scanner classes.	8L

	Reusability properties Super class & sub classes including multi level hierarchy ,process of constructor calling in inheritance ,use of super and final keywords with super() method, dynamic method dispatch, use of abstract classes & methods, interfaces. Creation of packages, importing packages, member access for packages.	4L
IV V	**Exception handling &Multi-threading** Exception handling basics, different types of exception classes, use of try &catch with throw, throws &finally, creation of user defined exception classes. Basics of multithreading, main thread, thread life cycle, creation of multiple threads, thread priorities, thread synchronization, inter-thread communication, deadlocks for threads, suspending &resuming threads.	8L
	Design pattern Design patterns. Introduction and classification. The iterator pattern Model-view-controller pattern. Commands as methods and as objects. Implementing OO language features. Memory management	4L
VI	**Graphical User interface :** Basics of applet programming, applet life cycle, difference between application & applet programming, parameter passing in applets, concept of delegation event model and listener, I/O in applets, use of repaint(),get Document Base(),get Code Base() methods, layout manager(basic concept), creation of buttons (JButton class only)& text fields. Graphical programming with Scale and Swing . The software development process **Generic types and collections** .	4L
	Abstract data types and their specification. How to implement an ADT. Concrete state space, concrete invariant, abstraction function. Implementing operations, illustrated by the Text exampl	4L

5.Text books

1. HerbertSchildt–"TheCompleteReference-Java2"–McGraw-Hill Pub.,5th Edition
2. E.Balagurusamy–"Programming withJava:APrimer"–McGraw-Hill Pub., 5th Edition
3. Malhotra Choudhary- "Programming in Java"- Oxford Pub., 1st Edition

REFERENCE BOOKS

1. Rambaugh, James Michael, Blaha – "Object Oriented Modelling and Design" – Prentice Hall, India
2. Ali Bahrami – "Object Oriented System Development" – Mc Graw Hill
3. Patrick Naughton, Herbert Schildt – "The complete reference-Java2" – TMH
4. R.K Das – "Core Java For Beginners" – VIKAS PUBLISHING
5. Deitel and Deitel – "Java How to Program" – 6th Ed. – Pearson
6. Ivor Horton's Beginning Java 2 SDK – Wrox
7. Spiegel M. R., Lipschutz S., John J.S., and Spellman D.: Complex Variables, TMH.

6.

Course Outcomes (CO):

Course Outcomes	**Details/Statement**	**Action Verb**	**Knowledge Level**
CO1	To elaborate the basic concept of object oriented programming and it's advantage over other approaches of programming.	Define, Recognize	Remember
CO2	Recognize features of object-oriented design such as encapsulation, polymorphism, inheritance, and composition of systems based on object identity	Explain	Understand Apply
CO3	Recognize features of object-oriented design such as Exception Handling and Multithreading in relation with some real life applications.	Illustrate	Understand Apply
CO4	Name and apply some common object-oriented design patterns and give examples of their use	Organize	Understand Analyze
CO5	. Design applications with an event-driven graphical user interface.	Determin e	Evaluate
CO6	Specify simple abstract data types and design implementations, using abstraction functions to document them	Identify, Implement	Apply

7. **Mapping of course outcomes to module / course content**

Module	CO1	CO2	CO3	CO4	CO5	CO6

8. **Mapping of the Course outcomes to Program Outcomes (PO)**

	PO1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO1 0	PO1 1	PO1 2
CO1												
CO2												
CO3												
CO4												
CO5												
CO6												

9. **Mapping to Program Specific Outcome (PSO)**

	PSO1	PSO2	PSO3	PSO4
CO1				
CO2				
CO3				
CO4				
CO5				
CO6				

***** End of Syllabus*****

Course Name: Computer organization and architecture Laboratory

Course Code: PCC-IT-491

(Semester- IV)

Course Broad Category: Information Technology

…………………………………………………………………………………………………

1. **Course Prerequisite:**
   The hardware-based design has been done in the Analog & Digital Electronics laboratory
   Course Learning Objectives:

   To gain practical knowledge by applying experimental methods to correlate with the theory.

   Teaching methodology and evaluation system for the course:
   Teaching methodology – Lectures and Presentations, Interactive Discussions.

Evaluation System–

Internal Assessment (60Marks)-Formative Continuous Assessment [Continuous Assessment; Note Book (30 Marks), Viva Voce (20 Marks), Attendance (10 Marks)]
End-Semester Exam (40 Marks)- Summative Assessment.

2. **Course Content:**
   Course Name: Computer organization and architecture
   Course Code: PCC-IT-491
   Hours per Week: 0L: 0T: 4P
   Credits: 2

Module	Topics	10P
1	Study of logic gates and realization of Boolean functions	1P
2	Implementation of half and full adder using simulator circuit	**1P**
3	Implementation of 2:4 Decoder and 4:2 Encoder	1P
4	Implementation of Multiplexer	1P
5	Implementation of composite adder and subtractor circuit	1P
6	Implementation of basic logic unit	1P
7	Implementation of arithmetic logic unit	1P
8	Implementation of BCD adder	1P
9	Understanding VHDL and implementation of OR gate using VHDL	1P
10	Implementation of various logic gate using VHDL	1P
11	Implementation of combinational circuits using VHDL	1P
12	Implementation of adder and subtractor VHDL programming	1P

5. References:

Text Book:

- M. Morris Mano, Rajib Mal—Computer system architecture; Publisher**. Pearson Education India.**
- Kai Hwang --- Advanced Computer architecture; **Publisher.** McGraw Hill Education**.**

Reference Books:

1. V. Carl, G. Zvonko and S. G. Zaky, “Computer organization”, McGraw Hill, 1978.
2. B. Brey and C. R. Sarma, “The Intel microprocessors”, Pearson Education, 2000.
3. J. L. Hennessy and D. A. Patterson, “Computer Architecture A Quantitative Approach”, Morgan Kauffman, 2011.
4. W. Stallings, “Computer organization”, PHI, 1987.
5. P. Barry and P. Crowley, “Modern Embedded Computing”, Morgan Kaufmann, 2012.
6. N. Mathivanan, “Microprocessors, PC Hardware and Interfacing”, Prentice Hall, 2004.
7. Y. C. Lieu and G. A. Gibson, “Microcomputer Systems: The 8086/8088 Family”, Prentice Hall India, 1986.
8. J. Uffenbeck, “The 8086/8088 Design, Programming, Interfacing”, Prentice Hall, 1987.
9. B. Govindarajalu, “IBM PC and Clones”, Tata McGraw Hill, 1991.
10. P. Able, “8086 Assembly Language Programming”, Prentice Hall India6. Winfried Karl Grassmann and Jean-Paul Tremblay, Logic and Discrete Mathematics, PEARSON.

6. Course Outcomes (CO):

Course Outcomes	**Details/Statement**	**Action Verb**	**Knowledge Level**
PCC-IT 491.1	Program implementation	Identify	Remember
PCC-IT 491.2	Analyze various components of circuit data movement.	Explain	Understand
PCC-IT 491.3	Logic gates implementation using circuit connection.	Implement	Apply
PCC-IT 491.4	Logic gates implementation using programming.	Organize	Analyze
PCC-IT 491.5	Implementation of adder/subtractor	Justify	Evaluate
PCC-IT 491.6	Various combinational circuit implementations.	Assess	Create

***** End of Syllabus*****

Course Name: Design and Analysis of Algorithms Lab
Course Code: PCC- IT-493
(Semester- IV)
Course Broad Category: PCC-MAJOR

……………………………………………………………………………………………………

1. Course Prerequisite:

- Basic knowledge of programming and general mathematical operations.

2. Course Learning Objectives:

I. **Develop problem-solving skills** by implementing basic C programming concepts, including variable types, type conversions, and arithmetic expressions in a structured programming environment.
II. **Apply control structures** such as branching statements (if-else, switch-case) and loops (while, do-while, for, nested loops) to solve computational problems efficiently.
III. **Implement data structures and functions** by working with arrays (1D & 2D), pointers, recursive functions, and string manipulations to enhance program modularity and efficiency.
IV. **Demonstrate file handling and structured data management** by using structures, unions, and file operations to develop small-scale projects like student and library information systems.

3. Teaching methodology and evaluation system for the course:

Teaching methodology – Lectures and Presentations, Interactive Discussions and Case Studies.

Evaluation System –
A. Internal Assessment (60 Marks)- Formative Continuous Assessment
B. End-Semester Exam (40 Marks)- Summative Assessment.

4. Course Content:

Course Name: Design and Analysis of Algorithms Lab
Course Code: PCC- IT-493
Hours per Week: 0L:0T: 4P
Credits: 2

Unit	Content

1	Implement Binary Search using Divide and Conquer approach Implement Merge Sort using Divide and Conquer approach
2	Implement Quick Sort using Divide and Conquer approach Find Maximum and Minimum element from array of integer using Divide and Conquer approach
3	Find the minimum number of scalar multiplication needed for chain of matrix
4	Implement all pair of Shortest path for a graph (Floyed- Warshall Algorithm) Implement Traveling Salesman Problem
5	Dynamic programming :Implement Single Source shortest Path for a graph (Dijkstra , Bellman Ford Algorithm
6	Brunch and Bound: Implement of Problem solving using Brunch and Bound
7	Backtracking: Implement N Queen problem, Graph Coloring Problem and Hamiltonian Problem
8	Greedy method Knapsack Problem ,Job sequencing with deadlines , Minimum Cost Spanning Tree by Prim's Algorithm Minimum Cost Spanning Tree by Kruskal'sAlgorithm
9	Graph Traversal Algorithm: Implement Breadth First Search (BFS) ,Implement Depth First Search (DFS)

5. References:

Text & References Books:

- Introduction to Algorithms, 4TH Edition, Thomas H Cormen, Charles E Lieserson, Ronald L Rivest and Clifford Stein, MIT Press/McGraw-Hill.
- Fundamentals of Algorithms – E. Horowitz et al.
- Algorithm Design, 1ST Edition, Jon Kleinberg and ÉvaTardos, Pearson.
- Algorithm Design: Foundations, Analysis, and Internet Examples, Second Edition, Michael T Goodrich and Roberto Tamassia, Wiley.
- Fundamentals of Computer Algorithms by Sartaj Sahni and Sanguthevar Rajasekaran Ellis Horowitz, University Press.
- Design & Analysis of Algorithms
  by Biswajit Bhowmik, Katson Publication.

6. Course Outcomes (CO):

After going through this course the Students will be able to:

Course Outcomes	Details	Action Verb	Knowledge Level
PCC-IT-494.CO1	Analyse different types of sorting and applications of Divide &Conquer techniques	Analyze	K4
PCC-IT-494.CO2	Understanding to implement Dynamic Programming techniques	Understand	K2
PCC-IT-494.CO3	Examine to implement knapsack, Job sequencing with deadlines, Prim’s and Kruskal’s algorithms by using greedy method	Examine	K4
PCC-IT-494.CO4	Discuss the implementation of the N-Queen and Graph Coloring Problem by using Backtracking	Discuss	K6
PCC-IT-494.CO5	Develop and implement problems by using Branch & Bound	Develop	K3

PCC-IT-494.CO6	Explain the way of implementation of BFS and DFS by using Graph Traversal Algorithms	Explain	K2

7. Mapping of course outcomes to module / course content

Unit	**CO 1**	**CO 2**	**CO 3**	**CO 4**	**CO 5**	**CO 6**
1						
2						
3						
5						
6						
7						

8. Mapping of the Course outcomes to Program Outcomes (PO)

	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1												
CO2												
CO3												
CO4												
CO5												
CO6												
AVG.												

9. Mapping to Program Specific Outcomes (PSO)

	PSO1	**PSO2**	**PSO3**	**PSO4**
CO1				
CO2				
CO3				
CO4				
CO5				
CO6				

***** End of Syllabus*****

DR. B.C. ROY SOCIETY
ENGINEERING
AN AUTONOMOUS INSTITUTE

Course Name: Object Oriented Programming Lab
Course Code: PCC-IT-495
(Semester –IV)
Course Broad Category: PCC Major

....

1. Course Prerequisite:

- Basic knowledge of computers and general mathematical operations.
- Basic Programming Knowledge

2. Course Learning Objectives:

1. **Encapsulation**: This principle involves bundling the data (attributes) and methods (functions) that operate on the data into a single unit, or object. This helps restrict direct access to some of the object's components, which can prevent unintended interference and misuse.
2. **Abstraction**: OOP allows programmers to reduce complexity by hiding the intricate details of the implementation and exposing only the necessary features of an object. This makes it easier to work with complex systems by focusing on the higher-level functionalities.

3. **Inheritance**: This feature allows a new class (subclass) to inherit properties and behaviors (methods) from an existing class (superclass). Inheritance promotes code reusability and establishes a natural hierarchy among classes.
4. **Polymorphism**: OOP enables objects to be treated as instances of their parent class, even when they are actually instances of a subclass. This allows for the implementation of methods that can operate on objects of different classes in a uniform way, enhancing flexibility and extensibility in code.
5. **Modularity**: OOP promotes the organization of code into distinct objects, making it easier to manage and maintain. This modularity allows developers to work on separate components of a program independently.
6. **Reusability**: By utilizing inheritance and encapsulation, OOP encourages the reuse of existing code, reducing redundancy and the potential for errors.
7. **Maintainability**: The structure provided by OOP makes it easier to update and maintain code. Changes in one part of the program can often be made with minimal impact on other parts.

These objectives make OOP a powerful approach for building complex software systems that are easier to understand, maintain, and extend.

3. Teaching methodology and evaluation system for the course:

Teaching methodology –Lectures and Presentations, Interactive Discussions and Case Studies.

Evaluation System –
A. Internal Assessment (60 Marks)- Formative Continuous Assessment
B. End-Semester Exam (40 Marks)- Summative Assessment.

4. Course Content:

Course Name: Object Oriented Programming Lab
Course Code: PCC-IT-495
Hours per Week: 0L: 0T: 4P
Credits: 2

Unit	Content
1	Assignments on class, constructor, overloading, inheritance, overriding
2	Assignments on wrapper class, arrays
3	Assignments on developing interfaces- multiple inheritance, extending interfaces
4	Assignments on creating and accessing packages
5	Assignments on multithreaded programming
6	Assignments on applet programming
	Note: Use Java for programming

5.Text books

1. HerbertSchildt–"TheCompleteReference-Java2"–McGraw-Hill Pub.,5thEdition
2. E.Balagurusamy–"Programming withJava:APrimer"–McGraw-Hill Pub., 5thEdition
3. Malhotra Choudhary- “Programming in Java”- Oxford Pub., 1stEdition

REFERENCE BOOKS

1Rambaugh, James Michael, Blaha – "Object Oriented Modelling and Design" – Prentice Hall, India
2 Ali Bahrami – "Object Oriented System Development" – Mc Graw Hill
3.Patrick Naughton, Herbert Schildt – "The complete reference-Java2" – TMH
4.R.K Das – "Core Java For Beginners" – VIKAS PUBLISHING
5.Deitel and Deitel – "Java How to Program" – 6th Ed. – Pearson
6.Ivor Horton's Beginning Java 2 SDK – Wrox
7 Spiegel M. R., Lipschutz S., John J.S., and Spellman D.: Complex Variables, TMH.

6 Course Outcomes (CO):

Course Outcomes	**Details/Statement**	**Action Verb**	**Knowledge Level**
CO1	To elaborate the basic concept of object oriented programming and it’s advantage over other approaches of programming.	Define, Recognize	Remember
CO2	Recognize features of object-oriented design such as encapsulation, polymorphism, inheritance, and composition of systems based on object identity	Explain	Understand Apply
CO3	Recognize features of object-oriented design such as Exception Handling and Multithreading in relation with some real life applications.	Illustrate	Understand Apply
CO4	Name and apply some common object-oriented design patterns and give examples of their use	Organize	Understand Analyze
CO5	. Design applications with an event-driven graphical user interface.	Determin e	Evaluate
CO6	Specify simple abstract data types and design implementations, using	Identify, Implement	Apply

	abstraction functions to document them		

8.Mapping of course outcomes to module / course content

Module	CO1	CO2	CO3	CO4	CO5	CO6

9.Mapping of the Course outcomes to Program Outcomes (PO)

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1												
CO2												
CO3												
CO4												
CO5												
CO6												

6. Mapping to Program Specific Outcome (PSO)

	PSO1	PSO2	PSO3	PSO4
CO1				
CO2				
CO3				
CO4				
CO5				
CO6				

***** End of Syllabus*****

Course Title: Technical Aptitude – I
Course Code: HSMC-491
Credits: 1 (0-0-2)
Semester: 4th Semester
Prerequisites: None

Course Objectives:

- To build foundational problem-solving abilities in quantitative and logical reasoning.
- To introduce basic programming aptitude and debugging techniques.
- To prepare students for technical screening rounds in internships and placements.
- To enhance analytical thinking aligned with Information Technology applications.

Course Outcomes (COs):
By the end of this course, students will be able to:

CO1: Solve basic quantitative aptitude problems involving arithmetic, algebra, and data interpretation.
CO2: Apply logical reasoning techniques to solve pattern recognition and analytical puzzles.
CO3: Understand and analyse simple programming logic using flowcharts or pseudocode.
CO4: Trace and predict outputs of basic code snippets in C/Python.
CO5: Identify and debug syntax and logical errors in elementary code segments.
CO6: Demonstrate improved speed, accuracy, and efficiency in technical aptitude tests.

List of Practical Sessions / Modules (Total: 12 sessions, 2 hrs/week)

Module	Topic	Activities
1	Introduction to Aptitude and Test Strategies	Orientation, types of questions, test formats
2	Number Systems & Simplifications	Practice problems, speed math techniques
3	Percentages, Profit & Loss	Real-life application problems, shortcuts
4	Time, Speed & Distance	Motion-based problems, unit analysis
5	Averages, Ratios & Proportions	Practice sets, group problem solving
6	Data Interpretation Basics	Tables, bar graphs, pie charts
7	Logical Reasoning – I	Series, patterns, odd one out
8	Logical Reasoning – II	Blood relations, directions, seating arrangements
9	Programming Aptitude – I	Pseudocode, flowcharts, dry runs
10	Programming Aptitude – II	Basic C/Python code tracing & output prediction
11	Debugging & Error Identification	Syntax/logic error spotting in simple code
12	Mini Test & Performance Feedback	Mock test (MCQs + Code Tracing) with discussion

---

Suggested Resources:

1. **Quantitative Aptitude for Competitive Examinations** – R.S. Aggarwal
2. **How to Prepare for Logical Reasoning for CAT** – Arun Sharma
3. **Programming Challenges** – Steven Skiena
4. Online platforms like HackerRank, CodeChef, GeeksforGeeks (for practice problems)

Course Outcomes (CO):

Course Outcomes	**Details / Statement**	**Action Verb**	**Knowledge Level**
HSMC-491.**CO1**	Solve basic quantitative aptitude problems involving arithmetic, algebra, and data interpretation.	Solve	Apply
HSMC-491.**CO2**	Apply logical reasoning techniques to solve pattern recognition and analytical puzzles.	Apply	Apply
HSMC-491.**CO3**	Understand and analyse simple programming logic using flowcharts or pseudocode.	Understand, Analyse	Understand, Analyse
HSMC-491.**CO4**	Trace and predict outputs of basic code snippets in C/Python.	Trace, Predict	Analyse
HSMC-491.**CO5**	Identify and debug syntax and logical errors in elementary code segments.	Identify, Debug	Evaluate
HSMC-491.**CO6**	Demonstrate improved speed, accuracy, and efficiency in technical aptitude tests.	Demonstrate	Apply

---

Mapping of Course Outcomes to Module / Course Content

Module	CO 1	CO 2	CO 3	CO 4	CO 5	CO 6
1	✓ Solve basic quantitative aptitude problems (CO1)	✓ Apply logical reasoning (CO2)	✗	✗	✗	✗
2	✓ Solve basic quantitative aptitude problems (CO1)	✗	✗	✗	✗	✓ Demonstrate improved speed, accuracy, and efficiency (CO6)
3	✓ Solve basic quantitative aptitude problems (CO1)	✗	✗	✗	✗	✓ Demonstrate improved speed, accuracy, and efficiency (CO6)
4	✓ Solve basic quantitative aptitude problems (CO1)	✗	✗	✗	✗	✓ Demonstrate improved speed, accuracy, and efficiency (CO6)
5	✓ Solve basic quantitative aptitude problems (CO1)	✓ Apply logical reasoning (CO2)	✗	✗	✗	✓ Demonstrate improved speed, accuracy, and efficiency (CO6)
6	✓ Solve basic quantitative	✗	✓ Understand and analyse simple	✗	✗	✓ Demonstrate improved

Module	CO 1	CO 2	CO 3	CO 4	CO 5	CO 6
	aptitude problems (CO1)		programming logic (CO3)			speed, accuracy, and efficiency (CO6)
7	✗	✓ Apply logical reasoning (CO2)	✗	✗	✗	✗
8	✗	✓ Apply logical reasoning (CO2)	✗	✗	✗	✗
9	✗	✗	✓ Understand and analyse simple programming logic (CO3)	✓ Trace and predict outputs of basic code snippets (CO4)	✗	✗
10	✗	✗	✓ Understand and analyse simple programming logic (CO3)	✓ Trace and predict outputs of basic code snippets (CO4)	✓ Identify and debug errors in elementary code (CO5)	✗
11	✗	✗	✗	✓ Trace and predict outputs of basic code snippets (CO4)	✓ Identify and debug errors in elementary code (CO5)	✗
12	✓ Solve basic quantitative aptitude	✓ Apply logical reasoning (CO2)	✓ Understand and analyse simple programming logic (CO3)	✓ Trace and predict outputs of basic code	✓ Identify and debug errors in elementary code (CO5)	✓ Demonstrate improved speed, accuracy, and

Module	**CO 1**	**CO 2**	**CO 3**	**CO 4**	**CO 5**	**CO 6**
	problems (CO1)			snippets (CO4)		efficiency (CO6)

Mapping of the Course outcomes to Program Outcomes

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
CO 1												
CO 2												
CO 3												
CO 4												
CO 5												
CO 6												

Mapping to Program Specific Outcome (PSO)

	POS 1	POS 2	POS 3	POS 4
CO 1				
CO 2				
CO 3				
CO 4				
CO 5				
CO 6				

===== THE END =====