
# SYLLABUS FOR COMPUTER SCIENCE (MAJOR)

Under Single Major Single Minor (FYUGP)

(To be implemented from Session 2024-25)

**SEM. I & II**

| Proposed Syllabus for four years B.Sc. Computer Science (Major) Programme | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Year** | **Semester** | **Paper Code** | **Paper** | **Credits** | **Periods/Week** | **Exam. Marks** | **Total Marks** |
| **1st Year** | **I** | COMSMAJ101 | Digital Design and Analysis | 3 | 3 | 60 | 80 |
| | | COMSMAJ101L | Digital Design and Analysis (Lab) | 1 | 4 | 20 | |
| | | COMSMAJ102 | Programming in C | 3 | 3 | 60 | 80 |
| | | COMSMAJ102L | Programming in C (Lab) | 1 | 4 | 20 | |
| | | POOASEC105<br>POOASEC106 | Basic Programming in Python<br>MS Power Point | 2 | 2 | 40 | 60 |
| | | POOASEC105L<br>POOASEC106L | Basic Programming in Python (Lab)<br>MS Power Point (Lab) | 1 | 2 | 20 | |
| | | MIN1 | Student has to choose only ONE discipline from the subjects given below:<br>1. Physics 6. Geography<br>2. Mathematics 7. Geology<br>3. Statistics 8. Microbiology<br>4. Chemistry 9. Zoology<br>5. Economics 10. Botany | 3 | 3 | 60 | 80 |
| | | MIN1 (L/T) | Minor Practical/Tutorial | 1 | 4 | 20 | |
| | | VAC | Environmental Education | 3 | 3 | 60 | 80 |
| | | VACT | Environmental Education (Tutorial) | 1 | 1 | 20 | |
| | **II** | COMSMAJ203 | Discrete Structures | 3 | 3 | 60 | 80 |
| | | COMSMAJ203T | Discrete Structures (Tutorial) | 1 | 1 | 20 | |
| | | COMSMAJ204 | Object Oriented Programming Using Java | 3 | 3 | 60 | 80 |
| | | COMSMAJ204L | Object Oriented Programming Using Java (Lab) | 1 | 2 | 20 | |
| | | POOBSEC218<br>POOBSEC219 | Cyber Security<br>HTML Programming | 2 | 2 | 40 | 60 |
| | | POOBSEC218L<br>POOBSEC219L | Cyber Security (Lab)<br>HTML Programming (Lab) | 1 | 1 | 20 | |
| | | MIN2 | Student will be provided the SAME discipline from the subjects selected previously as Minor. | 4 | 4 | 60 | 80 |
| | | MIN2 (L/T) | Minor Practical/Tutorial | 1 | 4 | 20 | |
| | | AEC1 | Compulsory English | 3 | 3 | 60 | 80 |
| | | AECT | Compulsory English (Tutorial) | 1 | 2 | 20 | |
| | | IDC1 | Student has to choose only ONE discipline from a pool of subjects that will be provided. | 2 | 2 | 40 | 60 |
| | | IDCT | Inter Disciplinary (Tutorial) | 1 | 1 | 20 | |
| | | IN1 | Summer Internship<br>The Colleges are expected to network with skill development centres, vocational training institutes for facilitating student internships. Online based internships programs are also permitted in case of Computer Science (Major) students. The students must submit a certificate of completion of the internship at the end of the semester. | 2 | - | - | |

NOTE: Tutorials should involve problem solving session/activity related to the subject taught.

**1st Year**
**Semester-I**

| | | | |
|---|---|---|---|
| **Course-MAJOR** | **Paper Code- COMSMAJ101** | **Credits-3** | **Lectures/Week-3** |
| **Paper:** | **Digital Design and Analysis** | | |

***Prerequisite(s) and/or Note(s):***
(1) High school Physics.
(2) Note(s): Syllabus changes yearly and may be modified during the term itself, depending on the circumstances. However, students will be evaluated only on the basis of topics covered in the course.

***Course Objectives***

***Knowledge acquired:***
(1) Basic knowledge of digital logic and digital circuits,
(2) Overall idea about how computers function and the internal building blocks of a computer.
(3) Knowledge about how operations are performed in a computer
(4) A thorough understanding of the fundamental concepts and techniques used in digital electronics.

***Skills gained:***
(1) Application of the knowledge of digital logic to understand digital electronics circuits.
(2) The ability to understand, analyze and design various combinational and sequential circuits.
(3) To understand and examine the structure of various number systems and its application in digital design.

***Competency Developed:***
(1) Ability to identify basic requirements for a design application and propose a cost effective solution.
(2) The ability to identify and prevent various hazards and timing problems in a digital design.
(3) Ability and skill to develop/build, and troubleshoot digital circuits.

**Syllabus Overview**

**Unit 1: Fundamentals of Computers** **7 Lectures**

Generation of Computers and Computer Languages, Computer Systems, Basic block Diagram, Von-Neumann Architecture, Types of Computers, Hardware, Firmware, I/O Devices, Storage classifications, Language translators.

**Unit 2: Number Systems and Codes** **10 Lectures**

Binary, octal, hexadecimal and decimal number systems and their inter conversion, BCD numbers (8421-2421), Gray code, excess–3 code, code conversion, ASCII, EBCDIC codes, their advantages and disadvantage, Binary addition and subtraction, Negative number representation: Sign magnitude, 1's, 2's Complement. signed and unsigned binary numbers, Fixed and floating-point representation.

**Unit 3: Logic Gates** **7 Lectures**

AND, OR, NOT Gates and their Truth Tables, NOR, NAND & XOR gates, Boolean algebra, Basic Boolean Laws, De-morgan's theorem, Boolean function and their truth tables, Minimization techniques, K-Map for 2, 3 and 4 variables, Sum of Product & Product of Sum, Don't care conditions.

**Unit 4: Logic Families** **7 Lectures**

Introduction to digital logic family such as RTL, DTL, TTL, ECL, CMOS, IIR, HTL etc., their comparative study, Basic circuit, performance characteristics.

**Unit 5: Combinational Logic** **7 Lectures**

Half adder, Full adder, parallel adder, half subtractor, full subtractor, 4-bit binary adder cum subtractor, Multiplexer, Demultiplexer, Decoder, BCD to seven segment Decoder, Encoders.

**Unit 6: Sequential Circuit:** **7 Lectures**

Set-reset latches, D-flip-flop, R-S flip-flop, J-K flip-flop, Master slave flip-flop, edge triggered flip-flop, T flip-flop, Synchronous/Asynchronous counter, Up/down synchronous counter, Ripple Counter, Applications of counter, Serial in/Serial out shift register, Parallel in/Serial out shift register, Serial in/parallel out shift register, parallel in/ parallel out shift register, Bi-directional register, Applications of register.

**Suggested Readings**

1. Rajaraman V. & Radhakrishnan, An Introduction To Digital Computer Design, PHI.
2. Malvino & Leach, Digital Principles & Applications, TMH
3. S. Salivahanan, S. Arivazhagan, Digital Circuits and Design, Oxford University Press

**Course-MAJOR Paper:** **Paper Code-COMSMAJ101L** **Credits-1** **Lab hours/Week-2**
**Digital Design and Analysis (Lab)**

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems:

1. Implementation of all the basic gates.
2. Design all gates using NAND gates.
3. Design all gates using NOR gates.
4. Design of a XOR gate using basic gates.
5. Design of an 8x1 MUX using basic gates.
6. Design of an 1x8 DEMUX using basic gates.
7. Design of a Decoder (different variants) using basic gates.
8. Design of an Encoder (different variants) using basic gates.
9. Simple Boolean Expression using Basic gates and Universal gates: (Examples)

   $A \cdot (\overline{B}+A) + B.\overline{A}$

   $XZ + X'YZ$

   $A + B[AC + (B + C')D]$

10. Design Half-Adder, Full-Adder, Half-Subtractor, Full-Subtractor Circuit

**Course-MAJOR Paper:** **Paper Code-COMSMAJ102** **Credits-3** **Lectures/Week-3**
**Programming in C**

***Prerequisite(s) and/or Note(s):***

(1) High school mathematics.
(2) Note(s): Syllabus changes yearly and may be modified during the term itself, depending on the circumstances. However, students will be evaluated only on the basis of topics covered in the course.

***Course Objectives***

***Knowledge acquired:***

(1) Knowledge about program development and implementation
(2) Syntax of C programming language
(3) Knowledge about how humans interact with computers through a language.

***Skills gained:***
(1) Problem solving skills
(2) Logical thinking to approach a problem
(3) Building programs for different problems at hand.

***Competency Developed:***
(1) Applying the skills learnt to model real world problems
(2) Facility in solving real life problems by thinking logically and outside of box.
(3) Ease of switching to any other programming language

**Syllabus Overview**

**Unit 1: Introduction to C, Data Types, Variables and Operators 6 Lectures**

History of C, Overview of Procedural Programming, Introduction to Algorithm & Flowcharts. Using main() function Compiling and Executing Simple Programs in C. Declaring, Defining and Initializing Variables, Scope of Variables, Using Named Constants, Keywords, Data Types, Casting of Data Types, Operators (Arithmetic, Logical and Bitwise), Using Comments in programs, Character I/O (getc, getchar, putc, putchar etc), Formatted and Console I/O (printf(), scanf()) , Using Basic Header Files (stdio.h, conio.h, stdlib.h, etc).

**Unit 2: Expressions, Conditional Statements and Iterative Statements 10 Lectures**

Simple Expressions in C (including Unary Operator Expressions, Binary Operator Expressions), Understanding Operators Precedence in Expressions, Conditional Statements (if construct, switch-case construct), Understanding syntax and utility of Iterative Statements (while, do-while, and for loops), Use of break and continue in Loops, Using Nested Statements (Conditional as well as Iterative)

**Unit 3: Understanding Functions 5 Lectures**

Utility of functions, Call by Value, Call by Reference, Functions returning value, Void functions, Return type of functions, Functions parameters, Differentiating between Declaration and Definition of Functions, Functions with variable number of Arguments.

**Unit 4: Implementation of Arrays and Strings 10 Lectures**

Creating and Using One Dimensional Arrays (Declaring and Defining an Array, Initializing an Array, Accessing individual elements in an Array, Manipulating array elements using loops), Use Various types of arrays (integer, float and character arrays / Strings) Two-dimensional Arrays (Declaring, Defining and Initializing Two Dimensional Array, Working with Rows and Columns). String handling functions.

**Unit 5: User-defined Data Types (Structures and Unions) 5 Lectures**

Understanding utility of structures and unions, Declaring, initializing and using simple structures and unions, Manipulating individual members of structures and unions, Array of Structures.

**Unit 6: Pointers and References in C 4 Lectures**

Understanding a Pointer Variable, Simple use of Pointers (Declaring and Dereferencing Pointers to simple variables)

**Unit 7: File and I/O 5 Lectures**

Opening and closing a file, Reading and writing Text Files, Using put(), get(), read() and write() functions, Random access in files.

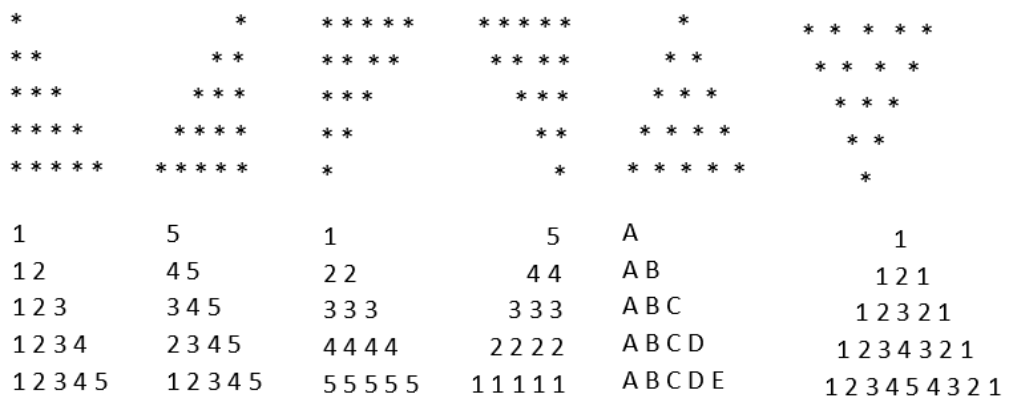
**Suggested Readings**


1. "The C Programming Language ANSI C Version", Kernighan & Ritchie, Prentice Hall Software Series
2. "ANSI C - Made Easy", Herbert Schildt, Osborne McGraw-Hill
3. "Learning to Program in C", N. Kantaris, Babani
4. "C - The Complete Reference", Herbert Schildt, Osborne McGraw-Hill
5. "Programming in C", Reema Thareja, Oxford University Press
6. "A First Course in Programming With C", T. Jeyapoovan, Vikas Publishing House
7. "Let Us C", Yashavant P. Kanetkar, BPB Publications


| **Course-MAJOR Paper:** | **Paper Code-COMSMAJ102L Programming in C (Lab)** | **Credits-1** | **Lab hours/Week-2** |
|---|---|---|---|

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems:

1. WAP to perform input/output of all basic data types.
2. WAP to enter two numbers and find their sum.
3. WAP to find the simple interest.
4. WAP to Swap Two Numbers (using and without using a third variable).
5. WAP to check whether a number is even or odd
6. WAP to compute the factors of a given number.
7. WAP to print the sum and product of digits of an integer.
8. WAP to check whether a character is vowel or consonant
9. WAP to find the largest among three numbers
10. WAP to compute the sum of the first 'n' terms of the following series
    S =1-2+3-4+5+…….n
11. WAP to compute the sum of the first 'n' terms of the following series
    S = 1+1/2+1/3+1/4+…….n
12. WAP to find the factors of a number.
13. WAP to print all the prime numbers within a given range
14. WAP to display the Fibonacci series.
15. WAP to find the factorial of a number.
16. WAP to check if a number is prime or not.
17. WAP to check if a number is Armstrong or not.
18. WAP to check if a number is Perfect or not.
19. WAP to print a triangle of stars as follows (take number of lines from user):



| Course- SEC | Paper Code-POOASEC105 | Credits-2 | Lectures/Week-2 |
|---|---|---|---|
| Paper: | Basic Programming in Python | | |

***Prerequisite(s) and/or Note(s):***

(1) High school mathematics.

(2) Note(s): Syllabus changes yearly and may be modified during the term itself, depending on the circumstances. However, students will be evaluated only on the basis of topics covered in the course.

***Course Objectives***

***Knowledge Acquired:***

(1) Fundamental Concepts: Students acquire knowledge of fundamental programming concepts such as variables, data types, loops, conditionals, and functions in Python.

(2) Data Structures: They learn about essential data structures like lists, tuples, dictionaries, and sets, understanding their usage and implementation.

***Skills Gained:***

(1) Coding Proficiency: Through hands-on practice and assignments, students develop coding proficiency in Python, enabling them to write clear, concise, and functional code.

(2) Problem-Solving: They enhance their problem-solving skills by applying Python programming concepts to solve various computational problems and algorithms.

(3) Debugging and Troubleshooting: Students acquire skills in debugging code and troubleshooting errors, learning how to identify and fix common programming mistakes effectively.

***Competency Developed:***

(1) Logical Thinking: Python programming exercises require logical thinking and algorithmic problem-solving skills, helping students develop a logical mindset.

(2) Attention to Detail: Writing code necessitates attention to detail to ensure accuracy and functionality. Students develop this competency through debugging and code review processes.

(3) Collaboration and Documentation: Students learn to collaborate on coding projects using version control systems like Git and to document their code effectively, enhancing their ability to work in teams and communicate technical concepts clearly.

**Syllabus Overview**

**Unit 1: Introduction to Python** **10 Lectures**

Structure of a Python Program, Elements of Python, Entering Expressions into the Interactive Shell, The Integer, Floating-Point, and String Data Types, String Concatenation and Replication, Storing Values in Variables

**Unit 2: Flow control and Functions** **10 Lectures**

Boolean Values, Comparison Operators, Boolean Operators, Mixing Boolean and Comparison Operators, Elements of Flow Control, Program Execution, Flow Control Statements, Importing Modules, Ending a Program Early with sys.exit(),def Statements with Parameters, Return Values and return Statements, The None Value, Keyword Arguments and print(), Local and Global Scope, The global Statement, Exception Handling.

**Unit 3: List, Dictionary, String and Tuples** **10 Lectures**

String, String functions, Manipulating Strings, Lists: Creating Lists; Operations on Lists; Built-in Functions on Lists; Implementation of Stacks and Queues using Lists; Nested Lists. Dictionaries: Creating Dictionaries; Operations on Dictionaries; Built-in Functions on Dictionaries; Dictionary Methods; Populating and Traversing Dictionaries. Tuples and Sets: Creating Tuples; Operations on Tuples; Built-in Functions on Tuples; Tuple Methods; Creating Sets; Operations on Sets; Built-in Functions on Sets; Set Methods.

**Suggested Readings**


1. T. Budd, Exploring Python, TMH, 1st Ed, 2011
2. Python Tutorial/Documentation www.python.or 2015
3. Allen Downey, Jeffrey Elkner, Chris Meyers , How to think like a computer scientist : learning with Python , Freely available online. 2012
4. http://docs.python.org/3/tutorial/index.html
5. http://interactivepython.org/courselib/static/pythonds
6. http://www.ibiblio.org/g2swap/byteofpython/read/


| **Course-SEC Paper:** | **Paper Code-POOASEC105L** | **Credits-1** | **Lab hours/Week-2** |
|---|---|---|---|
| | **Basic Programming in Python (Lab)** | | |

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems:

1. Write a menu driven program to convert the given temperature from Fahrenheit to Celsius and vice versa depending upon users' choice.
2. WAP to calculate total marks, percentage and grade of a student. Marks obtained in each of the three subjects are to be input by the user. Assign grades according to the following criteria:
    a. Grade A: Percentage >=80
    b. Grade B: Percentage>=70 and <80
    c. Grade C: Percentage>=60 and <70
    d. Grade D: Percentage>=40 and <60
    e. Grade E: Percentage<40
3. Write a menu-driven program, using user-defined functions to find the area of rectangle, square, circle and triangle by accepting suitable input parameters from user.
4. WAP to display the first n terms of Fibonacci series.
5. WAP to find factorial of the given number.
6. WAP to implement the use of arrays in Python.
7. WAP to implement String Manipulation in python in Python.
8. WAP to find sum of the following series for n terms: 1 – 2/2! + 3/3! - - - - - n/n!

| **Course- SEC Paper:** | **Paper Code- POOASEC105** | **Credits-2** | **Lectures/Week-2** |
|---|---|---|---|
| | **MS Power Point** | | |

***Prerequisite(s) and/or Note(s):***

(1) High school mathematics.

(2) Note(s): Syllabus changes yearly and may be modified during the term itself, depending on the circumstances. However, students will be evaluated only on the basis of topics covered in the course.

***Course Objectives***

***Knowledge Acquired:***

(1) Presentation design principles understanding.
(2) MS PowerPoint interface familiarity.
(3) Slide layout and formatting comprehension.

***Skills Gained:***

(1) Slide creation and editing proficiency.
(2) Visual content insertion capability.
(3) Animation and transition application skill.

***Competency Developed:***

(1) Effective presentation delivery competency.
(2) Audience engagement techniques mastery.
(3) Time management during presentations efficiency.

**Syllabus Overview**

| **Unit 1:** | **Creating and Managing Presentations** | **15 Lectures** |
|---|---|---|

Create a Presentation: Insert and Format Slides, Modify Slides, Handouts, and Notes, Change Presentation Options and Views, Configure a Presentation for Print, Configure and Present a Slide Show, Insert and Format Text: Insert and Format Shapes and Text Boxes, Insert and Format Images, Order and Group Objects.

| **Unit 2:** | **Tables, Charts, SmartArt, and Media** | **7 Lectures** |
|---|---|---|

Insert and Format Tables: Insert and Format Charts, Insert and Format SmartArt graphics, Insert and Manage Media.

| **Unit 3:** | **Transitions and Animations** | **8 Lectures** |
|---|---|---|

Apply Slide Transitions, Animate Slide Content, Set Timing for Transitions and Animations, Working with bullets and numbering, Working with different views, Working with slide Master, Slide show option

**Suggested Readings**


1.Microsoft power point 2019 ,learning the basics by Eric Stockson
2. Microsoft power point 2019 for beginners by J.Davidson.
3. Marquee series MIcrosoft power point 2019 by Audrey Roggenkamp & Lan Rutkowsky ,Nita Rutkosky


| **Course-SEC** | **Paper Code- POOASEC105L** | **Credits-1** | **Lab hours/Week-2** |
|---|---|---|---|
| **Paper:** | **MS Power Point (Lab)** | | |

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems:

(1) Creating a Title Slide
(2) Creating Slides Using Layouts
(3) Create a presentation that consists of 5 slides and save your Presentation in desktop.
(4) Demonstrate slide transitions and animation
(5) Insert slide number, slide date, slide header and footer
(6) Demonstrate rehearse time.
(7) Demonstrate master slide.

| 1st Year<br>Semester-II | | | |
|---|---|---|---|
| **Course-MAJOR** | **Paper Code- COMSMAJ203** | **Credits-3** | **Lectures/Week-3** |
| **Paper:** | **Discrete Structures** | | |

***Prerequisite(s) and/or Note(s):***
(1) High school Mathematics.
(2) Note(s): Syllabus changes yearly and may be modified during the term itself, depending on the circumstances. However, students will be evaluated only on the basis of topics covered in the course.

***Course Objectives***

***Knowledge acquired:***
(1) Basic knowledge of discrete mathematics and discrete structures,
(2) To develop understanding of Logic sets and functions
(3) Knowledge of mathematically correct terminology and notations.
(4) Knowledge about construction of direct and indirect proofs.

***Skills gained:***
(1) Development of problem-solving skills necessary for understanding counting problems.
(2) Ability to generalize from a single instance of a problem an entire class of problems and identification of patterns of data.

***Competency Developed:***
(1) Ability to analyse problems and solve problems.
(2) Ability to implement mathematical knowledge in data analysis.

**Syllabus Overview**

**Unit 1: Introduction** **10 Lectures**

Sets - finite and Infinite sets, uncountably Infinite Sets; functions, relations, Properties of Binary Relations, Closure, Partial Ordering Relations; counting - Pigeonhole Principle, Permutation and Combination; Mathematical Induction, Principle of Inclusion and Exclusion.

**Unit 2: Growth of Functions** **8 Lectures**

Asymptotic Notations, Summation formulas and properties, Bounding Summations, approximation by Integrals.

**Unit 3: Recurrence Relations** **10 Lectures**

Recurrence Relations, generating functions, Linear Recurrence Relations with constant coefficients and their solution,Substitution Method, Recurrence Trees, Master Theorem

**Unit 4: Graph Theory** **10 Lectures**

Basic Terminology, Models and Types, multigraphs and weighted graphs, Graph Representation, Graph Isomorphism, Connectivity, Euler and Hamiltonian Paths and Circuits, Planar Graphs, Graph Coloring, Trees, Basic Terminology and properties of Trees, Introduction to Spanning Trees, graph traversals (BFS, DFS).

**Unit 5: Propositional Logic** **7 Lectures**

Logical Connectives, Well-formed Formulas, Tautologies, Equivalences, Inference Theory, Quantifiers.

**Suggested Readings**

1.C.L. Liu , D.P. Mahopatra, Elements of Discrete mathematics, 2nd Edition , Tata McGraw Hill, 1985,
2.Kenneth Rosen, Discrete Mathematics and Its Applications, Sixth Edition ,McGraw Hill 2006
3.T.H. Coremen, C.E. Leiserson, R. L. Rivest, Introduction to algorithms, 3rd edition Prentice Hall on India, 2009
4.M. O. Albertson and J. P. Hutchinson, Discrete Mathematics with Algorithms , John wiley Publication, 1988
5.J. L. Hein, Discrete Structures, Logic, and Computability, 3rd Edition, Jones and Bartlett Publishers, 2009
6.D.J. Hunter, Essentials of Discrete Mathematics, Jones and Bartlett Publishers, 2008
7.Discrete Mathematical Structures with Applications to Combinatorics, V Ramaswamy, University Press
8.Discrete Mathematics: A Concept-based Approach, Basavaraj S Anami, Venkanna S Madalli, University Press

| **Course-MAJOR Paper:** | **Paper Code- COMSMAJ203T** **Discrete Structures (Tutorial)** | **Credits-1** | **Tut./Week-1** |
|---|---|---|---|

Discrete Structures Tutorial as assigned and advised by teacher(s).

| **Course-MAJOR Paper:** | **Paper Code- COMSMAJ204** **Object Oriented Programming Using Java** | **Credits-3** | **Lectures/Week-3** |
|---|---|---|---|

***Prerequisite(s) and/or Note(s):***
(1) High school mathematics.
(2) Note(s): Syllabus changes yearly and may be modified during the term itself, depending on the circumstances. However, students will be evaluated only on the basis of topics covered in the course.

***Course Objectives***

***Knowledge acquired:***
(1) Understanding of Object-Oriented Concepts: Students will acquire knowledge of fundamental Object-Oriented Programming (OOP) concepts such as classes, objects, inheritance, polymorphism, and encapsulation. They'll grasp the theoretical underpinnings of these concepts and their practical applications in software development.
(2) Java Syntax and Language Features: Through hands-on coding exercises and projects, students will become proficient in Java syntax, learning about data types, control flow structures, and exception handling. They'll understand how to write Java programs that follow best practices and adhere to industry standards.
(3) Software Design Principles: The course will equip students with knowledge of software design principles like SOLID principles, design patterns, and anti-patterns. They'll learn how to architect well-structured, maintainable, and extensible software systems using object-oriented design principles.

***Skills gained:***
(1) Programming Proficiency: Students will develop practical programming skills in Java, including the ability to write, compile, and execute Java programs independently. They'll gain confidence in coding by solving progressively challenging programming problems and implementing real-world applications.
(2) Debugging and Troubleshooting: Through debugging exercises and code reviews, students will learn how to identify and fix errors in Java code effectively. They'll develop skills in using debugging tools and techniques to diagnose and resolve software issues efficiently.

***Competency Developed:***

(1) Problem-Solving Skills: Students will enhance their problem-solving abilities by applying object-oriented principles to solve complex programming problems. They'll learn how to break down problems into smaller, manageable components and devise elegant solutions using OOP concepts.
(2) Critical Thinking and Analysis: The course will foster students' ability to critically evaluate software designs and code implementations. They'll learn to analyze trade-offs, identify design flaws, and propose alternative solutions, honing their critical thinking skills essential for software development.
(3) Software Development Practices: By working on practical projects, students will develop competency in software development practices such as unit testing, code documentation, and code refactoring. They'll understand the importance of writing clean, readable, and maintainable code, preparing them for careers in software engineering or further academic pursuits in computer science.

## Syllabus Overview

**Unit 1: Introduction to Java** **7 Lectures**

Java Architecture and Features, Compiling and Executing a Java Program, Variables, Constants, Keywords Data Types, Operators (Arithmetic, Logical and Bitwise) and Expressions, Comments, Doing Basic Program Output, Decision Making Constructs (conditional statements and loops) and Nesting, Java Methods (Defining, Scope, Passing and Returning Arguments, Type Conversion and Type and Checking, Built-in Java Class Methods).

**Unit 2: Arrays, Strings and I/O** **10 Lectures**

Creating & Using Arrays (One Dimension and Multi-dimensional), Referencing Arrays Dynamically, Java Strings: The Java String class, Creating & Using String Objects, Manipulating Strings, String Immutability & Equality, Passing Strings To & From Methods, String Buffer Classes. Simple I/O using System.out and the Scanner class, Byte and Character streams, Reading/Writing from console and files.

**Unit 3: Object -Oriented Programming Overview** **10 Lectures**

Principles of Object-Oriented Programming, Defining & Using Classes, Controlling Access to Class Members, Class Constructors, Method Overloading, Class Variables & Methods, Objects as parameters, final classes, Object class, Garbage Collection.

**Unit 4: Inheritance, Interfaces, Packages, Enumerations, Autoboxing and Metadata** **10 Lectures**

Inheritance: (Single Level and Multilevel, Method Overriding, Dynamic Method Dispatch, Abstract Classes), Interfaces and Packages, Extending interfaces and packages, Package and Class Visibility, Using Standard Java Packages (util, lang, io, net), Wrapper Classes, Autoboxing/Unboxing, Enumerations and Metadata.

**Unit 5: Exception Handling, Threading, Networking and Database Connectivity** **5 Lectures**

Exception types, uncaught exceptions, throw, built-in exceptions, Creating your own exceptions; Multi-threading: The Thread class and Runnable interface, creating single and multiple threads.

**Unit 6: Applets and Event Handling** **3 Lectures**

Java Applets: Introduction to Applets, Writing Java Applets, Working with Graphics.

**Suggested Readings**


1. Ken Arnold, James Gosling, David Homes, "The Java Programming Language", 4th Edition, 2005.
2. James Gosling, Bill Joy, Guy L Steele Jr, GiladBracha, Alex Buckley"The Java Language Specification, Java SE 8 Edition (Java Series)", Published by Addison Wesley, 2014.
3. Joshua Bloch, "Effective Java" 2nd Edition,Publisher: Addison-Wesley, 2008.
4. Cay S. Horstmann, GaryCornell, "Core Java 2 Volume 1 ,9th Edition,Printice Hall.2012
5. Cay S. Horstmann, Gary Cornell, "Core Java 2 Volume 2 - Advanced Features)", 9th Edition, Printice Hall.2013
6. Bruce Eckel, "Thinking in Java", 3rd Edition, PHI, 2002.
7. E. Balaguruswamy, "Programming with Java", 4th Edition, McGraw Hill.2009.
8. Paul Deitel, Harvey Deitel, "Java: How to Program", 10th Edition, Prentice Hall, 2011.


| **Course-MAJOR** | **Paper Code- COMSMAJ204L** | **Credits-1** | **Lab hours/Week-2** |
|---|---|---|---|
| **Paper:** | **Object Oriented Programming Using Java (Lab)** | | |

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems:

1. To find the sum of any number of integers entered as command line arguments
2. To find the factorial of a given number
3. To learn use of single dimensional array by defining the array dynamically.
4. To learn use of ".length" in case of a two dimensional array
5. To convert a decimal to binary number
6. To check if a number is prime or not, by taking the number as input from the keyboard
7. To find the sum of any number of integers interactively, i.e., entering every number from the keyboard, whereas the total number of integers is given as a command line argument

| **Course- SEC** | **Paper Code-POOBSEC218** | **Credits-2** | **Lectures/Week-2** |
|---|---|---|---|
| **Paper:** | **Cyber Security** | | |

***Prerequisite(s) and/or Note(s):***

(1) Anyone interested in learning about the subject.
(2) Note(s): Syllabus changes yearly and may be modified during the term itself, depending on the circumstances. However, students will be evaluated only on the basis of topics covered in the course.

***Course Objectives***

***Knowledge Acquired:***

(1) Cyber threats landscape understanding.
(2) Principles of cryptography comprehension.
(3) Network security protocols familiarity.

***Skills Gained:***

(1) Ethical hacking techniques application.
(2) Security assessment tools utilization.
(3) Incident response plan development.

***Competency Developed:***

(1) Risk assessment proficiency.
(2) Security policy formulation expertise.
(3) Communication of security concepts clarity

**Syllabus Overview**

**Unit 1: Introduction of Cyber Security 5 Lectures**

A Brief History of the Internet, Computer Crime, Defining Cyber Security and Cyberspace, Communication and web technology, Internet, World wide web, regulation of cyberspace, concept of cyber security, Issues and challenges of cyber security. Cyber security terminologies: Security, Attacks, risk, vulnerability, exploit, hacker, Computer Criminals, Cyber warfare, Security Services, Security Mechanisms, Case Studies.

**Unit 2: Cyber crimes 6 Lectures**

Cyber crimes targeting Computer systems and Mobiles - spyware, logic bombs, DoS, virus,Trojans, ransomware, data breach, Online scams and frauds - email scams, Phishing, Online job fraud, Online sextortion, Debit/ credit card fraud, Online payment fraud, website defacement,Cyber espionage, Darknet - illegal trades, drug trafficking, human trafficking, Social Media Scams & Frauds- impersonation, identity theft, misinformation, fake news , cyber crime against persons - cyber grooming, child pornography, cyber stalking, Cyber bullying, Social Engineering attacks, Crime reporting procedure, Case studies.

**Unit 3: Digital Devices Security, Tools & Technologies for Cyber Security 12 Lectures**

End Point device and Mobile phone security, Password policy, Security management, Data backup, Downloading and management of third party software, Device security policy, Cyber Security best practices, Significance of host firewall and Ant-virus, Wi-Fi security, Configuration of basic security policy and permissions.

**Unit 4: Cyber law and Investigation 7 Lectures**

Cyber crime and legal landscape around the world, IT Act, 2000 and its amendments. Limitations of IT Act, 2000. Cyber crime and punishments, Cyber Laws and Legal and ethical aspects related to new technologies- AI/ML, IoT, Blockchain, Darknet and Social media, Cyber Laws of other countries, Case Studies.

**Suggested Readings**


1. "Cybersecurity for Dummies" by Chey Cobb.
2. “Computer Hacking Beginners Guide” by Alan T. Norman
3. “Hacking: Computer Hacking, Security Testing, Penetration Testing, and Basic Security” by John Slavio.
4. Bharat Bhaskar, Electronic Commerce: Framework, Technology and Application, 4thEd., McGraw Hill Education
5. Security in Computing,3rd Edition,Charles P. Pfleeger & Shari Lawrence Pfeleeger,PHI
6. Cryptography and Network Security,A.Kahate,TMH


| **Course-SEC** | **Paper Code- POOBSEC218L** | **Credits-1** | **Tut. hours/Week-1** |
|---|---|---|---|
| **Paper:** | **Cyber Security (Lab)** | | |

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems:

1. What are the Roles and Responsibilities of System Administrator? Demonstrate the steps for creating the User account, setting permissions, and protecting your files with password.
2. What is Wifi? How do you configure the Wifi on Windows operating system
3. Write the steps for creating the User account, setting permissions and protecting your files with password.

4. Write the steps for installation of software from Open source Mode and Paid subscription mode.
5. Write the steps to establish peer to peer network connection using two systems in a LAN.
6. Write the steps in providing network security and to set Firewall Security in windows.
7. Prepare a Case study on Ransomware attacks.
8. Prepare a case study on Social Media Crime.
9. Write the steps to prevent the denial of Service attacks.
10. What is Malware? Write the steps to remove the malware from your PC.
11. List out the various Mobile security apps. Write the steps to install and use, one of the mobile security app.
12. Write the steps to analyze the E-Mail Application's security vulnerabilities.
13. Write the steps to read Email Headers and identify them as SPAM.
14. Write the steps to check whether the website is legitimate or not.
15. Write the steps to prevent the denial of Service attacks
16. Demonstrate the use of Network tools: ping, ipconfig, ifconfig, tracert, arp, netstat, whois
17. Demonstrate sending of a protected word document.

| **Course- SEC Paper:** | **Paper Code-POOBSEC219 HTML Programming** | **Credits-2** | **Lectures/Week-2** |
|---|---|---|---|

***Prerequisite(s) and/or Note(s):***
(1) Anyone interested in learning about the subject.

1. Experience with any text editor like notepad,
2. How to create directories and files on your computer.
3. How to navigate through different directories.
4. How to type content in a file and save them on a computer.
5. Understanding about images in different formats like JPEG, PNG format.

(2) Note(s): Syllabus changes yearly and may be modified during the term itself, depending on the circumstances. However, students will be evaluated only on the basis of topics covered in the course.

***Course Objectives***

***Knowledge Acquired:***
(1) Understanding of web page structure.
(2) Notepad, browser familiarity.
(3) Knowledge about building web pages.

***Skills Gained:***
(1) Proficiency in web page development.
(2) Visual content insertion capability.
(3) Creation of forms, tables.

***Competency Developed:***
(1) Effective web page development competency.
(2) Understanding the core concepts of web development and how web pages are constructed.
(3) Ability to structure and organize content on a web page effectively.
(4) Skills in creating forms to collect and manage user input.

**Syllabus Overview**

**Unit 1: HTML Overview 5 Lectures**

Introduction to web page designing using HTML: create and save an HTML document, access a web page using a web browser, Basic HTML document, HTML document structure.

**Unit 2: HTML basic tags 5 Lectures**

Heading tag, Paragraph tag, Line break tag, Centering content, Horizontal lines, Preserve formatting, HTML tags: html, head, title, body, (attributes: text, background, bgcolor, link, vlink, alink), br (break), hr (horizontal rule), inserting comments, h1..h6 (heading), p (paragraph), b (bold), i (italics), u (underline), ul (unordered list), ol (ordered list), and li (list item). Description lists: dl, dt and dd. Attributes of ol (start, type), ul (type).

**Unit 3: HTML Formatting 5 Lectures**

Bold Text, Italic Text, Underlined Text, Strike Text, Monospaced Font, Superscript Text, Subscript Text, Inserted Text, Deleted Text, Larger Text, Smaller Text, Grouping Content

**Unit 4: HTML – Images, Tables, Forms 15 Lectures**

Insert images: img (attributes: src, width, height, alt), sup (super script), sub(subscript). HTML Forms: Textbox, radio buttons, checkbox, password, list, combo box. Embed audio and video in a HTML page. Create a table using the tags: table, tr, th, td, rowspan, colspan

**Suggested Readings**


1. HTML 5.0 for Beginners, Vinod KumarMurugesan
2. Learn HTML in Easy way, Ritesh Kumar.
3. A Complete Reference, HTML and CSS, Thomas A. Powell


| **Course-SEC Paper:** | **Paper Code- POOBSEC219L HTML Programming (Lab)** | **Credits-1** | **Lab hours/Week-2** |
|---|---|---|---|

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems:

(1) Creating an HTML document for displaying a web page with the following tags:
   a. Bold
   b. Italic
   c. Alignment
   d. Paragraph
   e. Underline
   f. Text colour
   g. Background colour
   h. Heading
   i. Line break
   j. pre

(2) Design a web page of your CV with headings as objective, educational qualification, achievement, strength, hobbies and personal details.
   Apply the following specifications:
   a. Set any light colour as page background
   b. Bold and underline all headings
   c. Insert your image on the left side of the web page
   d. Use heading tag to specify the heading
   e. After every heading is over put a horizontal line
   f. Use pre tag for educational qualification.

(3) HTML program to create nested lists

(4) HTML program to create a form to take input from user and display it

(5) HTML program demonstrating use of tables.


# SYLLABUS FOR COMPUTER SCIENCE (MAJOR)

Under Single Major Single Minor (FYUGP)

(To be implemented from Session 2024-25)

**SEM. III & IV**

| Proposed Syllabus for four years B.Sc. Computer Science (Major) Programme | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Year** | **Semester** | **Paper Code** | **Paper** | **Credits** | **Periods/Week** | **Exam. Marks** | **Total Marks** |
| 2nd Year | III | COMSMAJ305 | Programming with C++ | 3 | 3 | 60 | 80 |
| | | COMSMAJ305L | Programming with C++ (Lab) | 1 | 4 | 20 | |
| | | COMSMAJ306 | Computer Architecture and Organization | 3 | 3 | 60 | 80 |
| | | COMSMAJ306L | Computer Architecture and Organization (Tutorial) | 1 | 4 | 20 | |
| | | POOCSEC332<br>POOCSEC333 | Fundamentals of Computer Networking<br>MS Excel | 2 | 2 | 40 | 60 |
| | | POO CSEC332L<br>POO CSEC333L | Fundamentals of Computer Networking (Lab)<br>MS Excel (Lab) | 1 | 2 | 20 | |
| | | MIN3 | Student will be provided the SAME discipline from the subjects selected previously as Minor. | 3 | 3 | 60 | 80 |
| | | MIN3 (L/T) | Minor Practical/Tutorial | 1 | 4 | 20 | |
| | | AEC1 | Any one from MIL (Bengali/Nepali/Hindi/Urdu/Sanskrit/Alternative English) | 3 | 3 | 60 | 80 |
| | | AEC1T | MIL (Tutorial) | 1 | 2 | 20 | |
| | | IDC2 | Student has to choose only ONE discipline from a pool of subjects that will be provided. | 2 | 2 | 40 | 60 |
| | | IDC2T | Inter Disciplinary (Tutorial) | 1 | 1 | 20 | |
| | IV | COMSMAJ407 | Operating System | 3 | 3 | 60 | 80 |
| | | COMSMAJ407T | Operating System (Lab) | 1 | 1 | 20 | |
| | | COMSMAJ408 | Data Structure | 3 | 3 | 60 | 80 |
| | | COMSMAJ408L | Data Structure (Lab) | 1 | 2 | 20 | |
| | | MIN4 | Student will be provided the SAME discipline from the subjects selected previously as Minor. | 4 | 4 | 60 | 80 |
| | | MIN4 (L/T) | Minor Practical/Tutorial | 1 | 4 | 20 | |
| | | VAC2 | Understanding India | 3 | 3 | 60 | 80 |
| | | VAC2T | Understanding India (Tutorial) | 1 | 2 | 20 | |
| | | IDC3 | Student has to choose only ONE discipline from a pool of subjects that will be provided. | 2 | 2 | 40 | 60 |
| | | IDC3T | Inter Disciplinary (Tutorial) | 1 | 1 | 20 | |
| | | IN1 | Summer Internship<br>The Colleges are expected to network with skill development centres, vocational training institutes for facilitating student internships. Online based internships programs are also permitted in case of Computer Science (Major) students. The students must submit a certificate of completion of the internship at the end of the semester. | 2 | - | - | |

NOTE: Tutorials should involve problem solving session/activity related to the subject taught.

**2nd Year**
**Semester-III**

| | | | |
|---|---|---|---|
| **Course-MAJOR** | **Paper Code- COMSMAJ305** | **Credits-3** | **Lectures/Week-3** |
| **Paper:** | **Programming with C++** | | |

***Brief Course Description:***
This course introduces the principles of object-oriented programming using C++.
It covers core concepts such as classes, objects, inheritance, and polymorphism.
Students will develop the ability to write efficient, modular, and reusable code using C++.

***Prerequisite(s) and/or Note(s):***

1. Basic knowledge of programming logic at the high school level.
2. Prior exposure to any programming language like C is helpful but not mandatory.
3. **Note(s):**
   - The syllabus is subject to minor modifications depending on institutional or academic requirements.
   - Students will be assessed only on the basis of topics covered in class during the semester.

***Course Objectives***

***Knowledge Acquired:***

1. Understand the fundamental principles of object-oriented programming (OOP).
2. Learn the syntax and semantics of the C++ programming language.
3. Develop a conceptual understanding of classes, objects, inheritance and polymorphism.
4. Learn how to implement modular, reusable, and maintainable code using OOP concepts.

***Skills Gained:***

1. Ability to write efficient, error-free C++ programs using object-oriented techniques.
2. Application of arrays, functions, classes, and inheritance to solve real-world problems.
3. Skills in operator overloading, polymorphism, and working with pointers and memory management.
4. Ability to use C++ constructs to design small-scale systems or components.

***Competency Developed:***

1. Ability to break down a problem into objects and define appropriate class structures.
2. Capability to handle complexity through encapsulation and modularity.
3. Confidence in handling object lifecycle (construction/destruction) and memory allocation.
4. Readiness for further studies in data structures, algorithms, and system-level programming using C++ or similar languages.

**Syllabus Overview**

| | | |
|---|---|---|
| **Unit 1:** | **Language Fundamental** | **10 Lectures** |

**Overview of OOP**: The Object-Oriented paradigm, Basic concepts of OOP, Benefits of OOP, Object oriented languages, Application of OOP
**Overview of C++**: Evolution of C++ from C, Structure of a C++ program Data Types: Built-in data types, User-defined data types, Derived data types, Keywords, Constants and Variables: symbolic constants, Dynamic initialization of variable, Reference variable. Operators and expressions, Input and output using cin, cout, Control Structures: if-else, nested if-else, while, do-while, for, break, continue, switch, goto statement.

**Unit 2: Arrays, Structure & Functions** **10 Lectures**

**Arrays:** Single and multi-dimensional arrays, Character arrays (C-style strings), Introduction to C++ String class
**Structures**: A Simple structure, defining a structure variable, Accessing structure's member, Enumeration data type.
**Function**: Function Declaration, Calling Function, Function Definition, Passing Arguments to function: Passing Constant, Passing Value, Reference Argument, Structure as argument, Default Argument. Returning values from function: Return statement, Returning structure variable, Return by reference, Overloaded Function, Inline Function.

**Unit 3: Object and Classes** **10 Lectures**

**Object and Class:** Defining the class and its member, Making an outside function inline, nesting of member function, array as class member, Difference between classes and structures.
**Memory allocation**: Memory allocation for objects, new and delete operator, static data member, static member functions, object as function argument.
**Constructor & Destructor**: Constructors and types of constructors (Default constructor, Copy Constructor, Parameterized constructor), Destructors.

**Unit 4: Pointers and Inheritance** **7 Lectures**

**Pointers:** Introduction, & and * operator, pointer to object, this pointer, pointer to derived class.
**Inheritance**: Inheritance- definition, concept, Types of Inheritance, Visibility modes- Public, Private, Protected, Virtual Base Class, Benefits of Inheritance.

**Unit 5: Polymorphism** **8 Lectures**

**Static Polymorphism:** Operator overloading, Operator keyword, overloading unary operators (++ (pre increment and post increment), –) using operator function, overloading binary operators (+, -, =, >=, <=, +=, <, >, []). Friend function, Friend class, overloading binary operators using friend function and without using friend function.
**Dynamic polymorphism**: Virtual function, Pure Virtual Function, Abstract class.

**Suggested Readings**


1. 1. Object Oriented Programming with C++ : E. Balagurusamy, The McGraw-Hill
2. Let Us C++: Yashwant Kanetkar, BPB Publications
3. The C++ Programming Language: Bjarne Stroustrup, Addision Wasley.
4. Object Oriented Programming in C++ : Robert Lafore, Galgotia Publications.


| **Course-MAJOR Paper:** | **Paper Code- COMSMAJ305L Programming with C++ (Lab)** | **Credits-1** | **Lab hours/Week-2** |
|---|---|---|---|

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems:

1. Write a program to display your name, age, and course using cout.
2. Write a program to calculate the area of a rectangle using user input.
3. Write a program to check whether a number is even or odd using if-else.
4. Write a program to find the largest of three numbers using nested if-else.
5. Write a program to print the multiplication table of a number using a for loop.
6. Write a program to find the sum of digits of a number using a while loop.
7. Write a program to find the maximum and minimum elements in an array.
8. Write a program to reverse an array.
9. Write a program to sort an array using bubble sort.

10. Write a program to check whether a string is a palindrome (use character array).
11. Write a program using a structure to store and display student details (name, roll, marks).
12. Write a function to find the factorial of a number using recursion.
13. Write a program to overload a function sum() to calculate sum of 2, 3, or 4 integers.
14. Write a program that passes a structure as a function argument and displays data.
15. Write a program to create a class Rectangle with data members length and breadth, and member functions to calculate area and perimeter.
16. Write a program to create a class Student with constructor and destructor that displays a message when invoked.
17. Write a program to demonstrate use of static data members and static functions in a class.
18. Write a program to pass an object as an argument to a function.
19. Write a program to demonstrate the use of pointers with variables and arrays.
20. Write a program to create a base class Person and a derived class Student. Use inheritance to input and display details.
21. Write a program to implement multilevel inheritance (e.g., Person → Employee → Manager).
22. Write a program to demonstrate use of this pointer.
23. Write a program to overload the + operator to add two complex numbers.
24. Write a program using virtual functions to demonstrate runtime polymorphism.
25. Write a program to create an abstract class Shape with a pure virtual function area(), and derive Circle and Rectangle from it.

| **Course-MAJOR** | **Paper Code-COMSMAJ306** | **Credits-3** | **Lectures/Week-3** |
|---|---|---|---|
| **Paper:** | **Computer Architecture and Organization** | | |

***Brief Course Description:***

The course in Computer System Architecture aims to provide an in-depth understanding of various architectural designs in computing systems. It explores design decisions, architectural trade-offs, and their impact on system performance. It prepares students to analyze and compare computing architectures for enhanced efficiency and performance.

***Prerequisite(s) and/or Note(s):***

1. Familiarity with fundamental concepts in digital electronics and logic design.
2. **Students who have not taken Digital Systems** will need to do additional background reading on:
   - Combinational and sequential circuits
   - Assembly language basics
3. **Note(s):**
   - Course content may be revised during the term to align with academic priorities.
   - Students will be evaluated only on the topics covered in class.

***Course Objectives:***

***Knowledge Acquired:***

1. Understanding the structure and operation of computer systems at the architectural level.
2. Knowledge of CPU organization, memory hierarchy, instruction sets, and data path design.
3. Ability to comprehend the performance impact of architectural choices.
4. Understanding the design principles of input/output systems, control units, and memory systems.

***Skills Gained:***

1. Ability to analyse the working of different components in a computer system.
2. Proficiency in tracing instruction execution through data path and control logic.
3. Skills in comparing and evaluating architectural models based on performance metrics.

***Competency Developed:***

1. Capability to design and evaluate simple CPU architectures.
2. Ability to apply knowledge of memory organization, cache systems, and pipelining for system optimization.
3. Competency in hardware-software interaction, enhancing understanding of how software affects system performance.
4. Readiness for roles in system architecture design, embedded systems, performance analysis, and hardware-software co-design.

**Syllabus Overview**

**Unit 1: Register transfer and microoperations 6 Lectures**

Register transfer, Arithmetic microoperations, logic microoperations, shift microoperations, Computer registers, Need of Registers, bus system, Interconnection Structures, Bus Interconnection.

**Unit 2: Basic Computer Organization and Design 7 Lectures**

Instruction set, timing and control, instruction cycle, hardwired instruction format, memory reference, register reference and input-output instructions. Design of basic computer, Interrupt.

**Unit 3: Central Processing Unit 10 Lectures**

Stack organization, microprogrammed control. Microprogrammed instruction formats, addressing modes, instruction codes, machine language, assembly language, design of CPU, RISC, CISC architectures.

**Unit 4: Computer Arithmetic 8 Lectures**

Addition and Subtraction, Multiplication algorithm (Boots), Division algorithm, Floating point arithmetic operations.

**Unit 5: Memory and Input-Output Organization 10 Lectures**

Memory hierarchy, Main memory, Cache memory, Virtual memory. Input/Output Operations, Serial and Parallel Data Transfers, Programmed I/O, Interrupt-Driven I/O, Direct Memory Access.

**Suggested Readings**


1. M. Mano, Computer System Architecture, Pearson Education 1992
2. A. J. Dos Reis, Assembly Language and Computer Architecture using C++ and JAVA, Course Technology,2004
3. W. Stallings, Computer Organization and Architecture Designing for Performance, 8thEdition, Prentice Hall of India,2009
4. M. M. Mano, Digital Design, Pearson Education Asia,2013
5. Carl Hamacher, Computer Organization, Fifth edition, McGraw Hill, 2012.


**Course-MAJOR Paper Code-COMSMAJ306L Credits-1 Tutorial hours/Week-2**
**Paper: Computer Architecture and Organization (Tutorial)**

Computer System Architecture tutorials as assigned and advised by teacher(s).

| Course- SEC | Paper Code-POOCSEC332 | Credits-2 | Lectures/Week-2 |
|---|---|---|---|
| Paper: | Fundamentals of Computer Networking | | |

***Prerequisite(s) and/or Note(s):***

1. Basic Computer Literacy: Students should be familiar with fundamental computer operations such as using the keyboard, mouse, files, folders, and basic software applications.
2. No Prior Networking Knowledge Required: This is an introductory course, designed for beginners.
3. Recommended for: Undergraduate students from Computer Science, IT, Electronics, or any discipline with interest in Information Technology.

***Course Objectives:***

1. To introduce students to the basic concepts and purpose of computer networks.
2. To explain the different types of networks, devices, transmission media, and communication protocols.
3. To provide a foundational understanding of network architectures such as the OSI and TCP/IP models.
4. To familiarize students with basic IP addressing, configuration of small networks, and networking tools.
5. To develop practical skills for designing, setting up, and troubleshooting basic wired and wireless networks.

***Knowledge Acquired:***

By the end of this course, students will have gained theoretical and practical understanding of:

1. The role and components of computer networks in modern communication.
2. Types of networks (LAN, WAN, MAN), topologies, and data transmission media.
3. Hardware components like routers, switches, hubs, access points, NICs, and their uses.
4. Networking protocols such as TCP/IP, HTTP, FTP, SMTP, DNS, and DHCP.
5. Network security fundamentals including firewall basics, encryption, and safe internet practices.

***Skills Gained:***

1. Identify and differentiate between types of networks, hardware, and topologies.
2. Setup, configure, and troubleshoot a basic wired or wireless network.
3. Use basic networking tools and commands like ping, ipconfig, tracert, and netstat.
4. Share resources like printers and files over a local network, Configure Wi-Fi routers and manage basic internet sharing settings.
5. Implement safe browsing and basic network security practices.

***Competency Developed:***

1. Understanding the foundational structure and operation of computer networks.
2. Performing hands-on tasks involved in configuring small network environments.
3. Communicating effectively using networking terminology and concepts.
4. Applying practical problem-solving techniques in network setup and troubleshooting.
5. Enhancing employability for roles such as IT support technician, helpdesk executive, or junior network administrator.

**Syllabus Overview**

**Unit 1: Introduction to Computer Networks 12 Lectures**

Definition and objectives of computer networks; Advantages and applications of networking; Types of networks: LAN, MAN, WAN, PAN; Network topologies: Bus, Ring, Star, Mesh, Hybrid; Modes of transmission: Simplex, Half Duplex, Full Duplex; Categories of networks: Peer-to-Peer vs. Client-Server; Basic network architecture: OSI Model (overview) and TCP/IP model (layers and functions)

**Unit 2: Networking Hardware and Transmission Media 6 Lectures**

Networking devices: NIC (Network Interface Card), Switch, Hub, Router, Bridge, Gateway, Repeater, Access Point; Transmission media - Guided: Twisted Pair, Coaxial Cable, Fiber Optic, Unguided -Wi-Fi, Bluetooth, Infrared, Satellite, Concept of signal types: Analog and Digital signals; IP addressing: IPv4 and IPv6 (Basics only)

**Unit 3: Network Protocols and Configuration 5 Lectures**

Concept of protocol and protocol stacks; Important networking protocols: TCP/IP, HTTP/HTTPS, FTP, SMTP, POP3, DNS and DHCP; IP Addressing schemes, subnetting (basic concepts); Understanding firewalls and antivirus for network protection.

**Unit 4: Network Setup, Troubleshooting, and Security Basics 7 Lectures**

Steps to create a small network (wired/wireless LAN setup); Configuring a Wi-Fi router and sharing Internet connection; File and printer sharing over LAN; Common network troubleshooting tools and commands: ping, ipconfig, tracert, nslookup, netstat; Introduction to cybersecurity basics in networking Safe browsing practices, securing Wi-Fi, network encryption (WPA/WPA2); Introduction to Virtual Private Networks (VPN)

**Suggested Readings**


1. Data Communications and Networking, Author: Behrouz A. Forouzan, Publisher: McGraw-Hill Education.
2. Computer Networks, Author: Andrew S. Tanenbaum & David J. Wetherall, Publisher: Pearson Education
3. Computer Networking: A Top-Down Approach, Author: James F. Kurose & Keith W. Ross Publisher: Pearson
4. Networking All-in-One For Dummies (8th Edition), Author: Doug Lowe, Publisher: Wiley
5. Introduction to Networking, Author: Bruce Hallberg, Publisher: McGraw-Hill


**Supplementary Materials and Online Resources:**

1. Cisco Networking Essentials – Craig Berg Hands-on book with basic configuration and troubleshooting for beginners.
2. CompTIA Network+ Guide to Networks – Jill West, Tamara Dean, Jean Andrews Good for learners aiming at industry certification and real-world examples.
3. Cisco Networking Academy – https://www.netacad.com Free structured courses by Cisco for beginner to intermediate networking students.
4. GCF LearnFree – Computer Networking – https://edu.gcfglobal.org/en/computers Interactive tutorials and beginner-level explanations of networking concepts.
5. Microsoft Learn – Networking Fundamentals – https://learn.microsoft.com

| Course-SEC | Paper Code-POOCSEC332L | Credits-1 | Lab hours/Week-2 |
|---|---|---|---|
| Paper: | Fundamentals of Computer Networking (Lab) | | |

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems:

1. Create a diagram illustrating LAN, MAN, and WAN with examples.
2. Identify and compare various network topologies: Star, Ring, Bus, Mesh.
3. Simulate peer-to-peer and client-server communication using two computers.
4. Categorize different types of networks based on geographic coverage and purpose.
5. Label and explain the seven layers of the OSI model using a real-world analogy.
6. Create a table comparing OSI vs TCP/IP model with examples for each layer.
7. Identify different networking devices (hub, switch, router, NIC, modem) and state their purpose.
8. Connect two computers using Ethernet crossover cable and test connectivity.
9. Demonstrate the use of a Wi-Fi dongle and access point to connect a PC wirelessly.
10. Create a visual layout of different network cables (coaxial, twisted pair, fibre optic) and describe their use.
11. Test signal strength in various areas using a Wi-Fi analyser tool.
12. Create a presentation showing the differences between analog and digital transmission signals.
13. Use ping to test the connectivity between two devices on a network.
14. Demonstrate tracert or traceroute to trace the route to a public domain (e.g., www.google.com).
15. Demonstrate the use of nslookup command to find IP addresses of domains.
16. Demonstrate the use of email protocols (SMTP/POP/IMAP) using a mail client.
17. Set up a small wired LAN for file sharing between 3 computers.
18. Install and configure a Wi-Fi router with password.
19. Enable and disable Windows firewall and observe its effect on ICMP (ping) traffic.
20. Use netstat to observe active connections on a system and interpret the results.

| Course- SEC | Paper Code- POOCSEC333 | Credits-2 | Lectures/Week-2 |
|---|---|---|---|
| Paper: | MS Excel | | |

***Prerequisite(s) and/or Note(s):***

1. Basic knowledge of using a computer and operating systems (Windows/macOS)
2. Familiarity with basic mathematical operations
3. No prior experience with MS Excel is required Course Objectives:
1. To provide comprehensive knowledge of Microsoft Excel from basic to advanced features
2. To enable students to efficiently manage, analyze, and visualize data
3. To develop problem-solving skills using formulas, functions, and tools in Excel
4. To prepare students for professional and academic tasks involving spreadsheet usage

***Knowledge Acquired:***

1. Understanding of the Excel interface and navigation
2. Concepts of workbook, worksheets, cell references, ranges, and data types
3. Knowledge of formatting, sorting, filtering, and conditional formatting
4. Deep understanding of formulas and functions (logical, statistical, lookup, text, date/time)
5. Knowledge of data visualization tools such as charts and graphs.

***Skills Gained:***

1. Creating, formatting, and managing worksheets and workbooks
2. Applying and combining functions for calculations and data analysis
3. Using charts and graphs to represent data visually and effectively
4. Collaborating and protecting workbooks with shared access and security features
5. Importing/exporting data from other sources

***Competency Developed:***

1. Ability to use MS Excel efficiently in academic, personal, and professional contexts
2. Proficiency in data management and decision-making based on spreadsheet analysis
3. Enhanced productivity through automation and advanced Excel tools
4. Readiness for administrative, business analytics, accounting, data entry, and reporting roles
5. Foundation for certification exams like Microsoft Office Specialist (MOS) – Excel

## Syllabus Overview

| **Unit 1:** | **Introduction to MS Excel** | **12 Lectures** |
|---|---|---|

Overview of spreadsheet software and applications; Getting started with Excel: interface, ribbons, toolbars, and menus; Creating, saving, and opening workbooks; Understanding cells, rows, columns, ranges ; Data entry: text, numbers, dates; Basic formatting: font, alignment, number formats, borders, and colours, Introduction to worksheet management (insert, delete, rename, move)

| **Unit 2:** | **Working with Formulas and Functions** | **7 Lectures** |
|---|---|---|

Cell referencing: relative, absolute, and mixed references; Basic arithmetic formulas and operators; Common Mathematical functions: SUM, AVERAGE, PERCENTAGE, MIN, MAX, and COUNT; Logical: IF, AND, OR; Text: CONCATENATE, LEFT, RIGHT, LEN, TRIM; Date/Time: TODAY, NOW, DAY, MONTH, YEAR

| **Unit 3:** | **Data Management and Presentation** | **5 Lectures** |
|---|---|---|

Sorting and Filtering data; Conditional Formatting, Data validation and drop-down lists, Charts and Graphs: Column, Bar, Line, Pie charts; Formatting chart elements; Printing worksheets and setting page layouts

| **Unit 4:** | **Practical Applications and Productivity Tools** | **5 Lectures** |
|---|---|---|

Using Templates in Excel; Working with Multiple Sheets; Introduction to Tables and basic data analysis; Freeze Panes, Split View, and Zoom features; Basic PivotTable creation and interpretation; Protecting worksheets and workbooks; Exporting data (PDF, CSV)

**Suggested Readings**


1. Microsoft Excel 2021 Step by Step , *Author:* Curtis Frye, *Publisher:* Microsoft Press
2. Excel 2021 Bible, *Author:* Michael Alexander, Richard Kusleika, *Publisher:* Wiley
3. Excel Basics In 30 Minutes (2nd Edition), *Author:* Ian Lamont, *Publisher:* i30 Media Corporation
4. Microsoft Excel Data Analysis and Business Modeling, *Author:* Wayne L. Winston *Publisher:* Microsoft Press
5. Excel All-in-One For Dummies, *Author:* Greg Harvey, *Publisher:* Wiley

| Course-SEC | Paper Code- POOCSEC333L | Credits-1 | Lab hours/Week-2 |
|---|---|---|---|
| Paper: | MS Excel (Lab) | | |

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems:

1. Create a new workbook and save it with the name Student_Info.xlsx.
2. Enter the details of 10 students: Name, Roll No, Class, and Marks in 3 subjects.
3. Format the worksheet by applying borders, font styles, and background color.
4. Rename the worksheet as “Student Data” and insert a header and footer.
5. Adjust column widths and row heights to fit the data properly.
6. Use the Wrap Text and Merge & Center options for a title row.
7. Copy data from one sheet to another and move a worksheet within the workbook.
8. Insert a new row and column in your data and input additional student information.
9. Calculate the Total Marks and Average Marks of each student using formulas.
10. Apply the IF function to assign grades (e.g., IF(Average ≥ 60, "Pass", "Fail")).
11. Use the MAX, MIN, and COUNT functions on marks.
12. Use the ROUND, ROUNDUP, and ROUNDDOWN functions on average marks.
13. Use TEXT functions like LEFT, RIGHT, MID, and CONCATENATE on student names.
14. Insert the current date using the TODAY() function and show how many days have passed since a given date using DATEDIF().
15. Apply Sorting to arrange the students in ascending order of total marks.
16. Apply Conditional Formatting to highlight marks below 40 in red.
17. Create a Pie Chart showing percentage-wise marks of a single student.
18. Freeze the top row and the first column to make headings visible while scrolling.
19. Prepare a print-friendly version of your worksheet using Page Layout settings.
20. Create a new sheet and apply a Template for a monthly expense tracker.
21. Create a PivotTable to summarize total marks by class.
22. Protect a worksheet with a password and try editing locked cells.
23. Use the Goal Seek tool to find out the marks required in one subject to achieve a total of 250.
24. Split the worksheet window and use Zoom to focus on a section of data.
25. Export the student data to PDF and CSV formats.

**2nd Year**
**Semester-IV**

| | | | |
|---|---|---|---|
| **Course-MAJOR** | **Paper Code- COMSMAJ407** | **Credits-3** | **Lectures/Week-3** |
| **Paper:** | **Operating System** | | |

***Prerequisite(s) and/or Note(s):***

1. Basic understanding of computer organization and architecture
2. Familiarity with data structures and fundamental programming concepts
3. Note(s): This course builds foundational knowledge essential for advanced topics in system-level programming and distributed computing.

***Course Objectives:***

***Knowledge Acquired:***

1. Understanding the fundamental principles and functions of modern operating systems
2. Ability to analyse and solve common problems related to process management, memory, storage, and device control
3. Appreciation of the role operating systems play in bridging computer hardware and user-level applications
4. Foundational knowledge that supports further studies in areas such as systems programming, distributed systems, and cloud computing

***Skills Gained:***

1. Proficiency in writing and understanding system-level code
2. Ability to manage memory, schedule processes, and understand file systems and I/O operations
3. Skills in debugging and performance analysis of operating system components
4. Enhanced problem-solving abilities relevant to concurrent programming and resource management

***Competency Developed:***

1. Ability to design, implement, and optimize operating system components
2. Understanding of security, access control, and basic networking from an OS perspective
3. Competency in solving real-world system-level problems involving concurrency, deadlocks, and scheduling
4. Readiness for roles in software development, system programming, system administration, cybersecurity, and research in OS and distributed systems

**Syllabus Overview**

**Unit 1: Introduction** **8 Lectures**

Basic OS functions, resource abstraction, types of operating systems, multi-programming systems, batch systems, time sharing systems, multiprocessing system, operating systems for personal computers & workstations, process control & real time systems.

**Unit 2: Operating System Organization** **5 Lectures**

Processor and user modes, kernels, system calls and system programs

**Unit 3: Process Management** **12 Lectures**

System view of the process and resources, process abstraction, process hierarchy, threads, threading issues, thread libraries; Process Scheduling, non-pre-emptive and pre-emptive scheduling algorithms

**Unit 4: Concurrency and Synchronization** **10 Lectures**

Process synchronization, critical section, semaphores, methods for inter-process communication. Deadlocks: System model, deadlock characterization, deadlock prevention, detection and avoidance, recovery from deadlock, banker's algorithm.

**Unit 5: Memory Management** **10 Lectures**

Physical and virtual address space; memory allocation strategies –fixed and variable partitions, demand paging, page replacement algorithms, allocation of frames, thrashing, segmentation, and virtual memory.

**Suggested Readings:**


1. A.Silberschatz, P.B.Galvin, G.Gagne, Operating Systems Concepts, 8thEdition, John Wiley Publications2008.
2. A.S.Tanenbaum, Modern Operating Systems, 3rd Edition, Pearson Education 2007.
3. G.Nutt, Operating Systems: A Modern Perspective, 2nd Edition Pearson Education1997.
4. W.Stallings, Operating Systems, Internals & Design Principles, 5th Edition, Prentice Hall of India.2008.
5. M.Milenkovic, Operating Systems- Concepts and design, Tata McGraw Hill1992.
6. Operating Systems, A.K Sharma, University Press.


| **Course-MAJOR Paper:** | **Paper Code- COMSMAJ407L Operating System (Lab)** | **Credits-1** | **Lab hours/Week-2** |
|---|---|---|---|

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems:

1. WAP in C for implementation of Priority scheduling.
2. WAP in C for implementation of preemptive and non-preemptive scheduling.
3. WAP in C for implementation of Round Robin scheduling.
4. WAP in C for implementation of FCFS scheduling.
5. WAP in C for implementation of SJF scheduling.
6. WAP in C for implementation of producer – consumer problem using semaphores.
7. WAP in C for implementation of Banker's algorithm for deadlock avoidance.
8. WAP in C for implementation of deadlock avoidance.
9. WAP in C for implementation of memory allocation methods for fixed partitioning (First fit)
10. WAP in C to create a child process using fork (), display parent and child process id. Child process should display the message "Child process" and the parent process should display the message "Parent process".
11. WAP in C to accept n integers to be sorted. Main function creates child process using fork system call. Parent process sorts the integers in ascending order and waits for child process using wait system call. Child process sorts the integers in descending order.
12. WAP in C to implement the process system calls.

| Course-MAJOR Paper: | Paper Code- COMSMAJ408 Data Structure | Credits-3 | Lectures/Week-3 |
|---|---|---|---|

***Prerequisite(s) and/or Note(s):***

1. Basic knowledge of the C or C++ programming language
2. Familiarity with fundamental programming concepts such as loops, functions, and arrays
3. Note(s): This course provides the foundation for advanced studies in algorithms, software development, and system design

***Course Objectives:***

***Knowledge Acquired:***

1. Ability to design and implement fundamental data structures such as arrays, linked lists, stacks, queues, trees, and graphs using C or C++
2. Understanding of performance analysis and trade-offs in choosing appropriate data structures for various problems
3. Preparation for more advanced coursework in algorithms, databases, and software engineering

***Skills Gained:***

1. Development of algorithmic thinking and structured problem-solving approaches
2. Enhanced analytical and critical thinking skills to select and apply suitable data structures
3. Proficiency in abstraction and modelling of real-world problems using data structures
4. Strengthened coding skills through implementation and debugging of data structures in C or C++

***Competency Developed:***

1. Efficient use of computational resources through appropriate data structure design
2. Enhanced memory management skills and understanding of dynamic allocation
3. Debugging proficiency and adaptability in solving both theoretical and practical programming challenges
4. Readiness for roles in software development, system optimization, algorithm design, and technical interviews

## Syllabus Overview

| Unit 1: | Introduction and Arrays | 4 Lectures |
|---|---|---|

Definition of data structure, need, operations on data structures. Single and Multi-dimensional Arrays, Insertion, and deletion operations.

| Unit 2: | Linked Lists | 10 Lectures |
|---|---|---|

Singly, Doubly and Circular Lists; Normal and Circular representation of Stack in Lists. Insertion and deletion operations on singly, doubly and circular lists.

| Unit 3: | Stacks and Queues | 12 Lectures |
|---|---|---|

Implementing single/multiple stack/s in an Array, (Array and Linked representation), Push, Pop operations, Prefix, Infix and Postfix expressions, Applications of stack; Evaluation of theses expressions, conversion of these Expressions from one to another, Limitations of Array representation of stack. Array and Linked representation of Queue, De-queue, Priority Queues, Operations of queue.

| Unit 4: | Trees | 12 Lectures |
|---|---|---|

Introduction to Tree as a data structure; General tree, Binary Trees, conversion of a general tree to a binary tree, traversal of a binary tree, Binary search tree, AVL tree.

**Unit 5: Sorting and Searching** **7 Lectures**

Linear Search, Binary Search, Comparison of Linear and Binary Search, Selection Sort, Bubble Sort, Insertion Sort, Comparison of Sorting Techniques

**Suggested Readings**


1. Seymour Lipschutz, "Data Structures with C", Tata McGraw Hill Education Pvt Ltd.
2. Yashvant Kanetkar, "Data structures through C", bpb
3. Deshpande Kakde, "C and Data structures", dreamtech
4. Aaron M. Tenenbaum, Moshe J. Augenstein, Yedidyah Langsam, "Data Structures Using C and C++",2nd Edition, PHI, 2009.
5. Robert L.Kruse, "Data Structures and Program Design in C++", Pearson,1999.
6. D.S Malik, Data Structure using C++, 2nd Edition, Cengage Learning,2010.


**Course-MAJOR Paper:** **Paper Code- COMSMAJ408L Data Structure (Lab)** **Credits-1** **Lab hours/Week-2**

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems in C or C++:

1. Write a program to search an element from a list. Give user the option to perform Linear or Binary search.
2. WAP to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.
3. Implement Linked List. Include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists.
4. Implement Doubly Linked List. Include functions for insertion, deletion and search of a number.
5. Implement Circular Linked List. Include functions for insertion, deletion and search of a number.
6. Perform Stack operations using Linked List implementation.
7. Perform Stack operations using Array implementation.
8. Perform Queues operations using Circular Array implementation.
9. Create and perform different operations on Double-ended Queues using Linked List implementation and array implementation.
10. WAP to calculate factorial and to compute the factors of a given no. (i)using recursion, (ii) using iteration
11. WAP to display Fibonacci series (i) using recursion,(ii)using iteration
12. WAP to reverse the order of the elements in the stack using additional stack.
13. WAP to reverse the order of the elements in the stack using additional Queue.
14. WAP to implement Diagonal Matrix using one-dimensional array.
15. WAP to implement Lower Triangular Matrix using one-dimensional array.
16. WAP to implement Upper Triangular Matrix using one-dimensional array.
17. WAP to implement Symmetric Matrix using one-dimensional array.


# SYLLABUS FOR COMPUTER SCIENCE (MAJOR)

Under Single Major Single Minor (FYUGP)

(To be implemented from Session 2024-25)

**SEM. V&VI**

| Proposed Syllabus for four years B.Sc. Computer Science (Major) Programme | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Year** | **Semester** | **Paper Code** | **Paper** | **Credits** | **Periods/Week** | **Exam. Marks** | **Total Marks** |
| **3rd Year** | **V** | COMSMAJ509 | Database Management System | 3 | 3 | 60 | 80 |
| | | COMSMAJ3509L | Database Management System (Lab) | 1 | 2 | 20 | |
| | | COMSMAJ510 | Software Engineering | 3 | 3 | 60 | 80 |
| | | COMSMAJ510T | Software Engineering (Tutorial) | 1 | 1 | 20 | |
| | | COMSMAJ511 | Data Communication and Computer Networks | 3 | 3 | 60 | 80 |
| | | COMSMAJ511T | Data Communication and Computer Networks (Tutorial) | 1 | 1 | 20 | |
| | | COMSMAJ512 | Python Programming | 3 | 3 | 60 | 80 |
| | | COMSMAJ512L | Python Programming (Lab) | 1 | 2 | 20 | |
| | **VI** | COMSMAJ613 | Theory of Computation | 3 | 3 | 60 | 80 |
| | | COMSMAJ613T | Theory of Computation (Tutorial) | 1 | 1 | 20 | |
| | | COMSMAJ614 | Design and Analysis of Algorithm | 3 | 3 | 60 | 80 |
| | | COMSMAJ614T | Design and Analysis of Algorithm (Tutorial) | 1 | 1 | 20 | |
| | | COMSMAJ615 | Microprocessor | 3 | 3 | 60 | 80 |
| | | COMSMAJ615L | Microprocessor (Lab) | 1 | 2 | 20 | |
| | | COMSMAJ616 | Computer Graphics | 3 | 3 | 60 | 80 |
| | | COMSMAJ616L | Computer Graphics (Lab) | 1 | 2 | 20 | |

NOTE:Tutorials should involve problem solving session/activity related to the subject taught.

**3rd Year**
**Semester-V**

| | | | |
|---|---|---|---|
| **Course-MAJOR Paper:** | **Paper Code-COMSMAJ509 Database Management System** | **Credits-3** | **Lectures/Week-3** |

***Brief Course Description:***

This course provides a comprehensive introduction to the principles, design, and implementation ofDatabaseManagementSystems(DBMS).Studentswilllearnthefundamentalconceptsofdatabaseorganization,datamodeling,andSQL(StructuredQueryLanguage).Thecourseemphasizesboththeoretical understanding and practical skills necessary for effective database design, implementation,andmanagement.

***Prerequisite(s) and/or Note(s):***

1. Basic understanding of:
    a. Computer systems
    b. Data structures and Databases
    c. Programming concepts
2. **Note(s):**The syllabus is subject to minor modifications depending on institutional or academic requirements.Students will be assessed only on the basis of topics covered in class during the semester.

***Course Objectives***

***Knowledge Acquired:***

Upon completion of the course, students will:

1. Understand the principles of relational database design.
2. Be able to implement and manage relational databases.
3. Gain practical knowledge of SQL for data querying and manipulation.
4. Learn about database optimization techniques.
5. Be prepared for roles in:
    - Database administration
    - Database development
    - Data-driven application development

***Skills Gained:***

1. Database design and schema creation
2. Proficiency in SQL for querying, updating, and managing data
3. Data modeling using Entity-Relationship (ER) diagrams
4. Query optimization for efficient data retrieval
5. Application of normalization techniques to database structures
6. Awareness of database security and access control measures

***Competency Developed:***

1. Database connectivity through APIs and tools
2. Implementing backup and recovery solutions
3. Performing database administration tasks
4. Understanding the basics of data warehousing
5. Introduction to data mining and knowledge discovery
6. Fundamentals of distributed databases and their management

## Syllabus Overview

**Unit 1: Introduction 10 Lectures**

Definition of DBMS, file processing system Vs DBMS, Limitation of file processing system, databaseusers, Characteristics of database,database components,database system architecture- 3-layerand 2-layers, data independence, data dictionary. View of Data – Data Abstraction – Instances and Schemasdiagrams.DBA-DatabaseAdministrator-rolesandresponsibilitiesof aDBA

**Unit 2: Database models 7 Lectures**

A brief overview of three models- Hierarchical model, Network model and relationalmodel,comparisonofthree models.

**Unit 3: DatabaseDesign,ER-Diagram 12 Lectures**

ER-Model,Constraints,ER-Diagrams,different types of entities, attributes, relationships,E. F. Codd'srules,Keys-Primary,Candidate,Composite, Alternate, Foreign, Super, Relational Schemas, Normalization: Lossless decomposition, Lossydecomposition,Functionaldependencies,NormalForms- (Up toBCNF-1NF,2NF,3NF,BCNF).

**Unit 4: TransactionProcessing 4 Lectures**

ACIDproperties,Concurrencycontrol.

**Unit 5: SQLandORACLE 12 Lectures**

Oracle commands, Oracle datatypes, operators- Arithmetic, Logical, Range searching, Pattern Matching,Oracle functions- Aggregate, Numeric, String, Conversion, Date and Time, Oracle table-Dual, Groupingdata-Concept of grouping, Group by Clause, Having clause Language-DDL, DML, DCL, SQL Functions-queriesandsub-queriesinSQL, ConstrainsandindexesinSQL.

**Suggested Readings**


1. R.Elmasri,S.B.Navathe,FundamentalsofDatabaseSystems6thEdition, PearsonEducation, 2010.
2. R.Ramakrishanan, J.Gehrke, DatabaseManagementSystems3rdEdition, McGraw-Hill,2002.
3. A.Silberschatz,H.F. Korth,S.Sudarshan, DatabaseSystemConcepts6thEdition, McGrawHill,2010.
4. R.Elmasri,S.B.NavatheDatabaseSystemsModels,Languages,DesignandapplicationProgrammi ng,6thEdition,PearsonEducation,2013.
5. IvanBayross, Teach YourselfSQL &PL/SQL UsingORACLE8i& 9iwithSQLJ,1st Edition, BPBPublications,2003


**Course-MAJOR Paper Code- COMSMAJ509L Credits-1 Lab hours/Week-2**
**Paper: Database Management System(Lab)**

Studentsareadvicetodolaboratory/practicalpracticenot limitedto,butincludingthefollowingtypesofproblems.

1. Demonstratetablecreation
2. INSERTdataintothetable
3. SELECTtherecordfromthetable
4. SELECTallrowsandallcolumnsformthetable
5. Demonstratefilteringtabledata–Selectedcolumns andallrows,selectedrows andallcolumns,selectedcolumnsandselectedrows,Renametable.
6. SELECTallrecordsfromthetableeliminating duplicaterows.

7. Filteringtabledata
8. Sortingdatainatable
9. Creatingatablefromaanothertable
10. Insertingdataintoatablefromanothertable.
11. Demonstratedeleteoperations
12. Updatingthecontents ofatable
13. Modifyingthestructureof table
14. Destroyingtables
15. Findoutthetablecreatedbydifferentusers
16. Demonstratedifferenttypesof dataconstraints
17. DemonstratedifferentoperatorsinSQL-Arithmetic, logical,
18. Demonstratepatternmatching,rangesearching
19. Demonstrateoracletable'DUAL'
20. Demonstratedifferent oraclefunctions.

| **Course-MAJOR Paper:** | **Paper Code-COMSMAJ510 Software Engineering** | **Credits-3** | **Lectures/Week-3** |
|---|---|---|---|

***Brief Course Description:***

This course provides a comprehensive introduction to the principles, methodologies, and practices ofsoftware engineering. Students will learn the entire software development lifecycle, from requirementsgathering to design, implementation, testing, deployment, and maintenance.

***Prerequisite(s) and/or Note(s):***

1. Basic understanding of programming concepts
2. Basic familiarity with software development principles and terminology
3. **Note(s):**Course content may be revised during the term to align with academic priorities.Students will be evaluated only on the topics covered in class.

***Course Objectives:***

***Knowledge Acquired:***

Upon completion of the course, students will:

1. Understand and apply software engineering principles.
2. Effectively manage the Software Development Life Cycle (SDLC).
3. Contribute to the development of high-quality software systems.
4. Be prepared for roles in software development, project management, and related fields.
5. Gain knowledge in key software engineering areas, including:
    - SDLC phases
    - System design principles
    - Software testing techniques
    - Software configuration management

***Skills Gained:***

1. Application of SDLC models in real-world projects
2. Designing structured and modular software systems
3. Performing software testing and quality assurance
4. Managing software versions and configurations
5. Documenting and communicating software requirements and specifications

***Competency Developed:***

1. Strong programming skills relevant to software development
2. Enhanced problem-solving abilities

3. Application of algorithmic thinking to software design
4. Ability to participate in and contribute to software development projects
5. Working knowledge of software maintenance and evolution
6. Understanding the importance of team collaboration and project management in software engineering.

**Syllabus Overview**

| **Unit 1:** | **Introduction** | **4 Lectures** |
|---|---|---|

Definition of software and software engineering, the Evolving Role of Software, Software Characteristics,Softwarequality,ChangingNatureofSoftware,Softwareprocess.

| **Unit 2:** | **Software Development Models** | **6 Lectures** |
|---|---|---|

Software Development Life Cycle (SDLC), Types of Software Models- Waterfall Model, Iterative Model,SpiralModel,PrototypeModel.CapabilityMaturityModel Integration(CMMI).

| **Unit 3:** | **SoftwareRequirementAnalysisandPlanning** | **10 Lectures** |
|---|---|---|

Software Requirement Analysis and Modeling Techniques, Software Requirement Specification (SRS)-NeedforSRS,CharacteristicsandComponentsofSRS,Factfindingtechniques,CostEstimation-COCOMOModel,DataFlowDiagram(DFD)andEntity-RelationshipDiagram (ERD),

| **Unit 4:** | **SoftwareProjectManagement-SPM** | **8 Lectures** |
|---|---|---|

Define SPM, Needs of SPM, Software Risks, Risk Identification, RMMM Plan, Software Quality ControlandQualityAssurance,ISO(InternationalStandardsOrganization)9000Certification,Software Metrics.

| **Unit 5:** | **Coding** | **4 Lectures** |
|---|---|---|

Define Coding, Goals of Coding, characteristics of Programming Language, Programming style, CodingStandards,CodingGuidelines.

| **Unit 6:** | **TestingStrategies&Tactics** | **13 Lectures** |
|---|---|---|

Software Testing Fundamentals-Test Plan, Test Execution, Test Review, Inspection and Auditing, Testingprinciples, software Verification and validation, Types of software testing - Black-Box Testing, White-BoxTesting and their type, Other testing- Regression Testing, Mutation Testing, Acceptance testing, Alphatesting,Betatesting.

**Suggested Readings**


1. R.S.Pressman,SoftwareEngineering: APractitioner'sApproach(7thEdition),McGraw-Hill, 2009.
2. P.Jalote,AnIntegrated ApproachtoSoftwareEngineering(2ndEdition),NarosaPublishingHouse,2003.
3. K. K. AggarwalandY. Singh, SoftwareEngineering(2ndEdition),NewAgeInternationalPublishers,2008.
4. I.SommerVille, SoftwareEngineering(8thedition),AddisonWesley, 2006.
5. D.Bell, SoftwareEngineeringforStudents(4thEdition), Addison-Wesley, 2005.R.Mall,FundamentalsofSoftwareEngineering(2ndEdition),Prentice-HallofIndia,2004


| **Course-MAJOR Paper:** | **Paper Code-COMSMAJ510T** | **Credits-1** | **Tutorial hours/Week-2** |
|---|---|---|---|
| | **Software Engineering (Tutorial)** | | |

Software Engineering tutorial as assigned and advised by teacher(s).

**Course- MAJOR Paper Code-COMSMAJ511 Credits-2 Lectures/Week-2**
**Paper: Data Communication and Computer Networks**

***Prerequisite(s) and/or Note(s):***

1. Basic Computer Literacy: Students should be familiar with fundamental computer operations
such as using the keyboard, mouse, files, folders, and basic software applications.
2. No Prior Networking Knowledge Required: This is an introductory course, designed for beginners.
3. Recommended for: Undergraduate students from Computer Science, IT, Electronics, or any discipline with interest in Information Technology.

***Course Objectives:***

1. To introduce students to the basic concepts and purpose of computer networks.
2. To explain the different types of networks, devices, transmission media, and communication protocols.
3. To provide a foundational understanding of network architectures such as the OSI and TCP/IP
models.
4. To familiarize students with basic IP addressing, configuration of small networks, and networking tools.
5. To develop practical skills for designing, setting up, and troubleshooting basic wired and wireless networks.

***Knowledge Acquired:***

By the end of this course, students will have gained theoretical and practical understanding of:
1. The role and components of computer networks in modern communication.
2. Types of networks (LAN, WAN, MAN), topologies, and data transmission media.
3. Hardware components like routers, switches, hubs, access points, NICs, and their uses.
4. Networking protocols such as TCP/IP, HTTP, FTP, SMTP, DNS, and DHCP.
5. Network security fundamentals including firewall basics, encryption, and safe internet practices.

***Skills Gained:***

1. Identify and differentiate between types of networks, hardware, and topologies.
2. Setup, configure, and troubleshoot a basic wired or wireless network.
3. Share resources like printers and files over a local network, Configure Wi-Fi routers and manage basic internet sharing settings.
4. Implement safe browsing and basic network security practices.

***Competency Developed:***

1. Understanding the foundational structure and operation of computer networks.
2. Performing hands-on tasks involved in configuring small network environments.
3. Communicating effectively using networking terminology and concepts.
4. Applying practical problem-solving techniques in network setup and troubleshooting.
5. Enhancing employability for roles such as IT support technician, helpdesk executive, or junior
network administrator.

**Syllabus Overview**

**Unit 1: DataCommunicationFundamentalsandTechniques 10 Lectures**

Introduction:DataCommunications,Networks,Categoriesofnetwork-LAN, MAN, WAN,ModesofCommunication-Simplex,HalfDuplex,FullDuplex,Networktopologies,InternetHistory,LayeredNetwork Architecture-OSI model andTCP/IP protocol suite. Data and Signals- Analog and Digital signals,ParallelandSerialtransmission,Transmissionimpairment-Attenuation,DistortionandNoise,Multiplexingtechniques- FDM,TDM, WDM,Transmission media - Guided: Twisted Pair, Coaxial Cable, Fiber Optic, Unguided -Wi-Fi, Bluetooth, Infrared, Satellite.

**Unit 2: NetworksSwitchingTechniquesandAccessmechanisms 5 Lectures**

Switching-Circuitswitching,Packetswitching;Messageswitching,ConnectionlessandConnection-orienteddatagramswitching.

**Unit 3: DataLinkLayerFunctionsandProtocol 7 Lectures**

Errordetectionanderrorcorrectiontechnique-HammingCode,CRC,Checksum.Flowcontrolmechanism-SlidingWindowprotocol, go-back-nARQ,stopandwaitARQ

**Unit 4: MultipleAccessProtocolandNetworks 7 Lectures**

Multiple Access Protocols: Radom Access - ALOHA, CSMA, CSMA/CD, CSMA/CA, Controlled Access -Reservation, Polling, Token Passing, Channelization - FDMA, TDMA, CDMA. Network devices- repeaters,hubs,switches, bridges,router andgateways;

**Unit 5: NetworksLayer& Transport Layer FunctionsandProtocols 10 Lectures**

Routing- Routingalgorithms,classesofIP, IPv4andIPv6 Addresses, IP Addressing schemes, subnetting (basic concepts);Transport services-Congestion control, Connection establishment and release- three- way handshaking,NetworkSecurity-Cryptography

**Unit 6: Overview ofApplicationlayerprotocol 6 Lectures**

OverviewofDNSprotocol,overviewofWWW, Important networking protocols: HTTP/HTTPS, FTP, SMTP, POP3, DNS and DHCP; Understanding firewalls and antivirus for network protection.

**Suggested Readings**


1. Behrouz A. Forouzan, “Data Communications and Networking”, 5th Edition, McGraw HillEducation,2013.
2. AndrewS.Tanenbaum,DavidJ.Wetherall,“ComputerNetworks”,5thEdition,PrenticeHall,2011.
3. Larry L. Peterson and Bruce S. Davie, “Computer Networks A System Approach”, 5th Edition,MKP,2012.
4. James F. Kurose, Keith W. Ross, “Computer Networking, A Top-Down Approach”, 5thEdition,Pearson,2012.
5. Dr.RakeshKumarMandal:Computer NetworksforStudents, FirstEdition,SPD,2018


**Course-MAJOR Paper Code-COMSMAJ511T Credits-1 Lab hours/Week-2**
**Paper: Data Communication and Computer Networks (Tutorial)**

Data Communication and Computer Networks tutorial as assigned and advised by teacher(s).

| Course-MAJOR | Paper Code-COMSMAJ512 | Credits-3 | Lectures/Week-3 |
|---|---|---|---|
| Paper: | Python Programming | | |

**Prerequisite(s) and/or Note(s):**

(1) High school mathematics.

(2) Note(s): Syllabus changes yearly and may be modified during the term itself, depending on the circumstances. However, students will be evaluated only on the basis of topics covered in the course.

**Course Objectives**

**Knowledge Acquired:**

1. Fundamental Concepts: Students acquire knowledge of fundamental programming concepts such as variables, data types, loops, conditionals, and functions in Python.
2. Data Structures: They learn about essential data structures like lists, tuples, dictionaries, and sets, understanding their usage and implementation.

**Skills Gained:**

1. Coding Proficiency: Through hands-on practice and assignments, students develop coding proficiency in Python, enabling them to write clear, concise, and functional code.
2. Problem-Solving: They enhance their problem-solving skills by applying Python programming concepts to solve various computational problems and algorithms.
3. Debugging and Troubleshooting: Students acquire skills in debugging code and troubleshooting errors, learning how to identify and fix common programming mistakes effectively.

**Competency Developed:**

1. Logical Thinking: Python programming exercises require logical thinking and algorithmic problem-solving skills, helping students develop a logical mindset.
2. Attention to Detail: Writing code necessitates attention to detail to ensure accuracy and functionality. Students develop this competency through debugging and code review processes.
3. Collaboration and Documentation: Students learn to collaborate on coding projects using version control systems like Git and to document their code effectively, enhancing their ability to work in teams and communicate technical concepts clearly.

**Syllabus Overview**

| Unit 1: | Introduction to Programming | 6 Lectures |
|---|---|---|

Concept of problem solving, Problem definition, Program design, Debugging, Types of errors in programming, Documentation. Flowcharts, algorithms, Types of programming paradigms – unstructured, procedure oriented, object oriented. Programming methodologies - top-down and bottom-up programming.

| Unit 2: | Introduction to Python | 12 Lectures |
|---|---|---|

Structure of a Python Program, Elements of Python, Entering Expressions into the Interactive Shell, The Integer, Floating-Point, and String Data Types, String Concatenation and Replication, Storing Values in Variables**.**

| Unit 3: | Flow control and Functions | 12 Lectures |
|---|---|---|

Boolean Values, Comparison Operators, Boolean Operators, Mixing Boolean and Comparison Operators, Elements of Flow Control, Program Execution, Flow Control Statements, Importing

Modules, Ending a Program Early with sys.exit(), def Statements with Parameters, Return Values and return Statements, The None Value, Keyword Arguments and print(), Local and Global Scope, The global Statement, Exception Handling.

**Unit 4: List, Dictionary, String and Tuples 15 Lectures**

String, String functions, Manipulating Strings, Lists: Creating Lists; Operations on Lists; Built-in Functions on Lists; Implementation of Stacks and Queues using Lists; Nested Lists. Dictionaries: Creating Dictionaries; Operations on Dictionaries; Built-in Functions on Dictionaries; Dictionary Methods; Populating and Traversing Dictionaries. Tuples and Sets: Creating Tuples; Operations on Tuples; Built-in Functions on Tuples; Tuple Methods; Creating Sets; Operations on Sets; Built-in Functions on Sets; Set Methods.

**Suggested Readings:**


1.Yashavant Kanetkar and Aditya Kanetkar , Let Us Python,BPB,3rd Edition.
2. Sheetal Taneja and Naveen Kumar, Python Programming- A modular approach with Graphics, Database,
3. Mobile and Web applications, Pearson, Sixteenth Impression-2023,
4. T. Budd, Exploring Python, TMH, 1st Ed, 2011
5. Python Tutorial/Documentation www.python.or 2015
6. Allen Downey, Jeffrey Elkner, Chris Meyers, How to think like a computer scientist: learning withPython , Freely available online.2012


**Course-MAJOR Paper:** **Paper Code- COMSMAJ512L Python Programming (Lab)** **Credits-1** **Lab hours/Week-2**

Studentsareadvisedto dolaboratory/practicalpractice notlimitedto,but includingthefollowing types of problems:

1. Write a menu driven program to convert the given temperature from Fahrenheit to Celsius and vice versa depending upon users' choice.
2. WAP to calculate total marks, percentage and grade of a student. Marks obtained in each of the three subjects are to be input by the user. Assign grades according to the following criteria:

Grade A: Percentage >=80
Grade B: Percentage>=70 and <80
Grade C: Percentage>=60 and <70
Grade D: Percentage>=40 and <60
Grade E: Percentage<40

3. Write a menu-driven program, using user-defined functions to find the area of rectangle, square, circle and triangle by accepting suitable input parameters from user.

4. WAP to display the first n terms of Fibonacci series.
5. WAP to find factorial of the given number.
6. WAP to implement the use of arrays in Python.
7. WAP to implement String Manipulation in python in Python.
8. WAP to find sum of the following series for n terms: 1 – 2/2! + 3/3! - - - - - n/n!
9. WAP to calculate the sum and product of two compatible matrices.
10. WAP to create Class and Objects in Python.
12. WAP to implement Data Hiding in Python.
13. WAP to implement constructor and destructor for a class in Python.

14. WAP to implement constructor and destructor in Python.
15. WAP to implement different types of inheritance in Python.
16. WAP to implement concept of Overriding in Python.
17. Write programs to create mathematical 3D objects using class.
    a. curve b. sphere c. cone d. arrow e. ring f. cylinder
18. WAP to Check if a Number is Positive, Negative or 0
19. WAP to Check leap year or not.
20. WAP to display calendar of the given month and year.

**3rd Year**
**Semester-VI**

| | | | |
|---|---|---|---|
| **Course-MAJOR** | **Paper Code-COMSMAJ613** | **Credits-3** | **Lectures/Week-3** |
| **Paper:** | **Theory of Computation** | | |

**BriefCourseDescription:**

The Theory of Computation course explores into the foundational aspects of computer science, focusing on the study of formal languages, automata, and Turing machines. It explores the theoretical underpinnings of what can be computed and how efficiently computations can be performed. This course is crucial for understanding the limits of computation and the computational complexity of various problems.

**Prerequisite(s)and/orNote(s):**

Students interested in this course should have completed introductory courses in Discrete Mathematics and Algorithms. A solid grasp of mathematical logic and the ability to construct and understand mathematical proofs are essential for success in this course.

**CourseObjectives:**

The primary objectives of the Theory of Computation course are to impart a deep understanding of the basic concepts of computation, to familiarize students with various computational models, and to enable them to analyse the complexity of computational problems. By the end of the course, students will be well-versed in the theoretical aspects of computer science that are fundamental to advanced study and research in the field.

**Knowledgeacquired:**

1. Gain a comprehensive understanding, including finite automata, context-free grammars, and pushdown automata.
2. Learn about the hierarchy of formal languages and their corresponding automata models.
3. Study the theoretical framework of Turing machines and their role in computational theory.
4. Acquire knowledge about the boundaries of computation and the inherent limitations of computational processes.

**Skillsgained:**

Students will acquire skills to design and analyze various types of automata and grammars.
Develop problem-solving abilities related to computational complexity and algorithmic challenges.
Gain proficiency in constructing and proving theorems in the context of automata theory and formal languages.
These skills are essential for addressing complex problems in computer science and for conducting theoretical research.

**CompetencyDeveloped:**

1. Students will develop competence in identifying and solving intricate computational problems.
2. The subject will hone abstract thinking and formal reasoning abilities to apply theoretical concepts effectively.
3. Apply theoretical knowledge to real-world scenarios in fields such as algorithm design, software development, and computer science research.
4. This competency prepares students for advancing in diverse fields where complex problem-solving and theoretical understanding are crucial.

**Syllabus Overview**

| **Unit 1:** | **Languages** | **8 Lectures** |
|---|---|---|

Alphabets, string, language, Basic Operations on language, Chomsky hierarchy, Concatenation, Kleene Star

| **Unit 2:** | **Finite Automata and Regular Languages** | **15 Lectures** |
|---|---|---|

Regular Expressions, Transition Graphs, Deterministic and non-deterministic finite automata, NFA to DFA Conversion, Regular languages and their relationship with finite automata, Pumping lemma and closure properties of regular languages.

| **Unit 3:** | **Context free languages** | **12 Lectures** |
|---|---|---|

Context free grammars, parse trees, ambiguities in grammars and languages, Pushdown automata (Deterministic and Non-deterministic), Pumping Lemma, Properties of context free languages, normal forms.

| **Unit 4:** | **Turing Machines and Models of Computations** | **10 Lectures** |
|---|---|---|

Turing Machine as a model of computation, Universal Turing Machine, Language acceptability, recursively enumerable and recursive languages.

**Suggested Readings:**


1. Daniel I.A.Cohen, Introduction to computer theory, John Wiley,1996
2. Lewis & Papadimitriou, Elements of the theory of computation , PHI 1997.
3. Hoperoft, Aho, Ullman, Introduction to Automata theory, Language & Computation –3rd Edition, Pearson Education. 2006
4. P. Linz, An Introduction to Formal Language and Automata 4th edition Publication Jones Bartlett, 2006


| **Course-SEC Paper:** | **Paper Code-COMSMAJ613T Theory of Computation (Tutorial)** | **Credits-1** | **Lab hours/Week-2** |
|---|---|---|---|

Theory of Computation tutorial as assigned and advised by teacher(s).

**3rd Year**
**Semester-VI**

**Course-MAJOR Paper Code-COMSMAJ614 Credits-3 Lectures/Week-3**
**Paper: Design and Analysis of Algorithm**

**BriefCourseDescription:**

Design and Analysis of Algorithms is a core subject in computer science that focuses on developing efficient procedures for solving computational problems and assessing their performance. It involves crafting algorithms using techniques like divide and conquer, dynamic programming, and greedy methods, and evaluating their efficiency through time and space complexity using Big O notation. The subject also includes optimizing algorithms for better performance and exploring complexity classes, such as P and NP, to understand computational limits. Mastery of these concepts is essential for creating effective and efficient software and systems.

**Prerequisite(s)and/orNote(s):**

Students interested in this course should have had anexposure to introductory courses on programming and data structures. A basic of mathematical logic is essential for success in this course**.**

**CourseObjectives:**

The objective of learning Design and Analysis of Algorithms is to develop efficient methods for solving computational problems and to evaluate their performance using time and space complexity. It aims to equip students with skills to optimize algorithms for better speed and resource usage while understanding problem complexity and classification. This foundational knowledge is crucial for advancing in computer science and creating effective software solutions.

**Knowledgeacquired:**

1. Understanding how to measure and improve the efficiency of algorithms.
2. Proficiency in analyzing and comparing time and space complexity.
3. Skills to enhance algorithms for better performance and lower resource consumption.
4. Mastery of various algorithmic techniques such as divide and conquer, dynamic programming, and greedy methods.

**Skillsgained:**

1. Ability to conceptualize and develop step-by-step procedures for solving complex problems.
2. Proficiency in analyzing the time and space complexity of algorithms to gauge their efficiency.
3. Skills in refining algorithms to enhance their performance and minimize resource usage.
4. Mastery of various Problem-Solving Techniques, such as divide and conquer, dynamic programming, and greedy strategies.
5. Insight into the classification of problems and algorithms based on complexity classes like P, NP,etc.

**CompetencyDeveloped:**

1. Ability to evaluate and compare the efficiency of different algorithms using time and space complexity metrics.
2. Proficiency in refining and optimizing algorithms to improve performance and resource utilization.
3. Expertise in applying various algorithmic strategies and techniques to effectively solve diverse computational problems.
4. Enhanced capability to analyze and reason about the computational limits and trade-offs of different algorithmic approaches.

**Syllabus Overview**

| **Unit 1:** | **Introduction** | **6 Lectures** |
|---|---|---|

Definitions, Algorithms vs Programs, Basic AlgorithmDesignTechniques,CorrectnessofAlgorithm,Algorithm classification: Iterativetechniques,DivideandConquer,Dynamic Programming,GreedyAlgorithms, Algorithm analysis (formal and informal), calculation of running time of an algorithm, loop invariants.

| **Unit 2:** | **Growth of Functions** | **7 Lectures** |
|---|---|---|

Complexity of Algorithms, Best, Worst and Average case complexity, growth rate of functions, Asymptotic Analysis, Analyzing algorithm control structures.

| **Unit 3:** | **Recurrences** | **8 Lectures** |
|---|---|---|

Introduction, Substitution method, Iteration method, Master method (Master theorem), Simple problems using Master theorem.

| **Unit 4:** | **Analysis of simple SortingandSearchingalgorithms** | **16 Lectures** |
|---|---|---|

Elementary sorting techniques: Bubble Sort, Selection sort, Insertion Sort, Shell sort, Merge Sort, Advanced Sorting techniques - Heap Sort,QuickSort,SortinginLinearTime-BucketSort,RadixSortandCountSort,SearchingTechniques.

| **Unit 5:** | **Graphs** | **8 Lectures** |
|---|---|---|

GraphAlgorithms–BreadthFirst Search,DepthFirst SearchanditsApplications, Directed acyclic graphs, Single source shortest paths: Dijkstra's algorithm, MinimumSpanningTrees algorithms: Prim's algorithm, Kruskal's Algorithm.

**Suggested Readings**


1. T.H.Cormen,CharlesE.Leiserson,RonaldL.Rivest,CliffordSteinIntroductionto Algorithms,PHI,3rdEdition 2009
2. Sarabasse& A.V. Gelder Computer Algorithm – Introduction to Design and Analysis, Publisher Pearson 3rdEdition 1999
3. RajeshK.Shukla,AnalysisandDesignofAlgorithms:ABegineersApproach,Wiley
4. Udit Agarwal, Algorithms design and analysis, Dhanpat rai & Co.


| **Course-SEC Paper:** | **Paper Code-COMSMAJ614T** **Design and Analysis of Algorithms (Tutorial)** | **Credits-1** | **Lab hours/Week-2** |
|---|---|---|---|

Design and Analysis of Algorithms tutorial as assigned and advised by teacher(s).

**3rd Year**
**Semester-VI**

| **Course-MAJOR Paper:** | **Paper Code-COMSMAJ615** **Microprocessor** | **Credits-3** | **Lectures/Week-3** |
|---|---|---|---|

**BriefCourseDescription:**

The Microprocessor Fundamentals and Programming course provides a thorough introduction to the architecture, functioning, and programming of microprocessors. Students will explore the internal workings of microprocessors, learning how they interact with memory and peripherals to perform various tasks. This course emphasizes both theoretical concepts and practical programming skills, enabling students to develop a solid understanding of microprocessor-based systems.

**Prerequisite(s)and/orNote(s):**

Students should have a background in basic electronics and digital logic before enrolling in this course. Familiarity with assembly language or a high-level programming language is advantageous. This course is ideal for those pursuing studies in computer engineering, electrical engineering, or related fields.

**CourseObjectives:**

The primary objectives of the Microprocessor Fundamentals and Programming course are to provide students with a comprehensive understanding of microprocessor architecture and operation. The course aims to teach students how to program microprocessors in assembly language, interface with various hardware components, and design simple microprocessor-based systems. By the end of the course, students will be equipped with the knowledge and skills to develop and troubleshoot microprocessor applications.

**Knowledgeacquired:**

1. Students will gain detailed knowledge of the CPU, memory hierarchy, and input/output mechanisms.
2. They will learn about instruction sets, addressing modes, and the execution of instructions.
3. The course covers the principles of assembly language programming, enabling students to write and understand low-level code.
4. Students will learn about interfacing microprocessors with external devices, such as sensors and actuators.

**Skillsgained:**

1. Students will develop practical skills in programming microprocessors using assembly language.
2. They will learn how to write efficient code to control hardware, manage data, and perform arithmetic and logic operations.
3. Students will gain experience in debugging and optimizing assembly programs.
4. They will acquire skills in designing and implementing interfaces between the microprocessor and peripheral devices.

**CompetencyDeveloped:**

1. Students will have developed the competency to design, program, and troubleshoot microprocessor-based systems.
2. They will be capable of developing assembly language programs to solve complex problems and interface with various hardware components.
3. This competency will enable students to work in fields such as embedded systems, hardware design, and computer engineering.
4. The course provides a strong foundation for advanced studies in microprocessor technology and related areas.
5. Students will be prepared for careers in the rapidly evolving field of computing & electronics

## Syllabus Overview

| **Unit 1:** | **Microprocessor 8085 Architecture** | **15 Lectures** |
|---|---|---|

Introduction to Microprocessor – Introduction to Microprocessors, Components of

Microprocessor: Registers, ALU, timing and control, CPU, I/O devices, clock, memory, bussed architecture, tri-state logic, Bus Structure, address bus, data bus and control bus. Block diagram of 8085, pin configuration of 8085, Architecture of Microprocessor 8085, Internal registers (8-bit & 16-bit), CPU, ALU, multiplexing and demultiplexing address/data bus, Instruction Register and Decoder, Timing and Control Unit, Interrupts and Serial I/O.

**Unit 2: Instruction Set-I** **10 Lectures**

Machine Language and Assembly Language, Addressing modes, types of instruction format, Data Transfer type instructions, Arithmetic and logical instructions, Branching instructions -looping, Timing diagram for opcode fetch, memory read, memory write, I/O read, I/O write.

**Unit 3: Instruction Set-II and Programming** **10 Lectures**

Special Instructions: Rotate instructions - stack and subroutine related instructions. Assembly Language Programs – Addition, Subtraction, Multiplication (8-bit), Division (8-bit), sorting- Ascending / Descending Order, Largest/Smallest (single byte), Addition of N numbers (single byte).

**Unit 4: Memory/IO Interface and Interrupts** **10 Lectures**

Memory Interface (Basics) – memory mapped I/O & I/O mapped I/O. Interrupts in 8085- Types, Generation of RST codes-Hardware, software interrupts and their function, Edge triggered and level triggered interrupts, Interrupt priority, Vectored and non-vectored interrupt -SIM and RIM instructions, Comparison of Microprocessor and Microcontroller.

**Suggested Readings**


1. Microprocessor Architecture, Programming and Application with the8085, Ramesh S. Gaonakar, Penram International Publishing, Mumbai, (2011).
2. Fundamental of Microprocessor 8085: Architecture Programming, and Interfacing, V. Vijayendran, Viswanathan, S., Printers & Publishers Pvt. Ltd (2009).
3. The 8051 Microcontroller, Architecture, Program and application, Kenneth J Ayala, Pen ram
4. Microprocessor Organisation and Architecture, Leventhal L.A , Prentice Hall India.
5. B. Ram, Fundamentals of microprocessors and microcomputers – Dhanpat Rai Publications, New Delhi
6. The 8080/85 Family: Design, Programming & Interfacing, John Uffenbeck, , PHI India.
7. A. K. Ray & K. M.Bhurchandani, Advance Microprocessor and Peripherals, 2ndEdition, Tata McGraw Hill, 2006
8. Mathur A.P., Introduction to Microprocessors. 3rd edn, Tata McGraw, New Delhi,
9. Muhammed Ali Mazidi, Janice Gillispie Mazidi – The 8051 Microcontroller and Embedded systems
10. Microprocessors & Microcontrollers by B. P. Singh, Galgotia publications Pvt. Ltd.


| **Course-SEC Paper:** | **Paper Code-COMSMAJ615L Microprocessor (Lab)** | **Credits-1** | **Lab hours/Week-2** |
|---|---|---|---|

Students are advised to do laboratory/practical practice not limited to, but includingthe following types of programs

ASSEMBLYLANGUAGE PROGRAMMINGin 8085

1. Writean assemblylanguageprogramto checkwhethera numberiseven orodd.
2. Writean assemblylanguageprogramto findthe numberofones andthenumberof zerosin an8-bit
   number.
3. Writean assemblylanguage programto findthesmaller oftwonumbers.
4. Writeanassemblylanguage programtomultiplytwo8-bit numbers.
5. Write an assembly language program to find the smallest among 10 integers stored in memory
   locationsstartingfrom2050H
6. Write an assembly language program to find the largest among 10 integers stored in memory
   locations startingfrom2050H
7. Writeanassemblylanguageprogramtosort10numbers usingbubble sort.
8. Write an assembly language program to generate the first 10 fibonacci series and store the result at
   memorylocationstartingfromFC50H
9. Writean assemblylanguage programto findthesumof the series 12+22 +32 +42 + …..+ 102
10. Write an assembly language program to find the perfect square of any number and if the number
    is not aperfectsquare,displayFFH
11. Writean assemblylanguage programforlinearsearch.
12. Writean assemblylanguage programto calculatethefollowingexpression usinga singleregister:
    Y=X2+2X+3XZ
13. Writeanassemblylanguageprogramtofindthesumofthe first 10evennaturalnumbers.
14. Writeanassemblylanguage programtofindthesumof thefirst n oddnaturalnumbers.
15. Writean assemblylanguage programto create anoddparitygenerator.
16. Writeanassemblylanguageprogramtocreate anevenparitygenerator.
17. Writean assemblylanguage programtofindthesumof five 8-bit numbers.
18. Writeanassemblylanguageprogramto convert decimal tobinary.
19. Writeanassemblylanguageprogramtoconvertoctal tobinary.
20. Writeanassemblylanguageprogramtoconvert hexadecimaltobinary.
21. Writeanassemblylanguageprogramto convert hexadecimal todecimal.
22. Writeanassemblylanguageprogramtocheckwhetheran8-bitnumber ispalindrome or not.
23. Writeanassemblylanguage programtodisplaythetruthtable forandANDgate.
24. Writeanassemblylanguageprogramtodisplaythetruthtable forand ORgate.
25. Writeanassemblylanguageprogramto displaythetruthtable forandXORgate.
26. Writeanassemblylanguageprogramtoperformnbyteadditionoftwonumbers.
27. Writeanassemblylanguage programtoimplementasimplesubroutinecall.
28. Writean assemblylanguageprogramto checkwhethera number isprimeornot

**3rd Year**
**Semester-VI**

| | | | |
|---|---|---|---|
| **Course-MAJOR Paper:** | **Paper Code-COMSMAJ616 Computer Graphics** | **Credits-3** | **Lectures/Week-3** |

**BriefCourseDescription:**

Computer Graphics is a field of study that explores the creation, manipulation, and representation of visual images using computers. The course covers foundational concepts such as raster graphics, vector graphics, and image processing techniques. Students learn about algorithms and techniques for rendering 2D and 3D graphics, including shading, lighting, and texture mapping. The course also delves into graphical user interface (GUI) design, animation principles, and virtual reality applications. Practical skills in programming graphics applications and using relevant software tools

are emphasized, preparing students for careers in game development, animation, simulation, and visual effects industries.

**Prerequisite(s)and/orNote(s):**

To enroll in this course, students should have a basic understanding of several key areas of mathematics and geometry. This course is suitable for beginners and those looking to expand their knowledge in graphics and designing.

**CourseObjectives:**

The computer graphics course aims to teach students fundamental principles and techniques for creating and manipulating graphical images using computers. It covers algorithms for rendering 2D and 3D graphics, including shading and texture mapping, while emphasizing practical programming skills. Students gain expertise applicable to fields like animation, virtual reality, user interfaces, and interactive simulations, preparing them for careers in game development, animation, and visual effects.

**Knowledgeacquired:**

1. Understanding of fundamental principles and techniques for creating and manipulating graphical images.
2. Knowledge of algorithms for rendering 2D and 3D graphics, including shading, lighting, and texture mapping.
3. Practical skills in programming graphical applications and implementing rendering algorithms.

**Skillsgained:**

1. Proficiency in programming graphical applications and implementing rendering algorithms.
2. Skills in applying rendering techniques such as shading, lighting, and texture mapping to create realistic images.
3. Competence in designing intuitive user interfaces and interactive graphical elements.
4. Capability to solve complex visual problems and effectively communicate concepts through computer-generated imagery.

**CompetencyDeveloped:**

1. Gain proficiency in using graphics software and programming languages to create and manipulate graphical images.
2. Develop skills in designing and integrating visual elements into applications, games, or simulations.
3. Apply problem-solving skills to solve challenges related to visual representation and user interaction.
4. Acquire a foundational understanding of graphics principles and their applications in various domains, such as animation, virtual reality, and user interfaces.

## Syllabus Overview

**Unit 1: Introduction 4 Lectures**

Definition, Basic elements of Computer Graphics, Applications of Computer Graphics, Graphics pipeline, Types of Computer Graphics (Raster and Vector Graphics), Display resolution, aspect ratio and pixel concepts.

**Unit 2: Graphics Hardware 7 Lectures**

Architecture of Raster and Random scan display devices, input/output devices, Frame buffer and refresh rate, Display technologies (CRT, LCD, LED and OLED), Introduction to Graphics

Processing Unit (GPU), Interactive graphics systems.

**Unit 3: Fundamental Techniques in Graphics 15 Lectures**

Raster scan line, Circle and ellipse drawing, Thick primitives, Polygon filling, Line and polygon clipping algorithms, 2D and 3D Geometric Transformations, Composite transformations, Homogeneous coordinate system, 2D and 3D Viewing Transformations (Projections- Parallel and Perspective), Window-to-Viewport transformation, Vanishing points, Character generation and text display, Introduction to Anti-Aliasing.

**Unit 4: Geometric Modelling 7 Lectures**

Representing curves and surfaces, Parametric representation of curves, Bezier curves, B-Spline curves, Polygon mesh representation, Basic surface representation techniques.

**Unit 5: Visible Surface Determination & Surface Rendering 12 Lectures**

Hidden surface elimination, Back-face detection, Z-buffer method, Scan-line method, Painter's algorithm. Illumination and shading models, Basic color models (RGB, CMY and HSV), Texture mapping concepts, Computer Animation, Key frame animation, Introduction to Ray Tracing and Rendering.

**Suggested Readings**


1. J D Foley, A. Van Dan, Feiner, Hughes Computer Graphics Principles & Practice 2nd edition
Publication Addison Wesley 1990.
2. D. Hearn, Baker: Computer Graphics, Prentice Hall of India 2008.
3. D F Rogers Procedural Elements for Computer Graphics, McGraw Hill 1997.
4. D F Rogers, Adams Mathematical Elements for Computer Graphics, McGraw Hill 2nd edition
1989.
5. Salini Govil-Pai, Principles of Computer Graphics, University Press


**Course-MAJOR Paper:** **Paper Code-COMSMAJ616L** **Credits-1** **Lab hours/Week-2**
**Computer Graphics (Lab)**

Students are advised to do laboratory/practical practice not limited to, but including the following types of programs

1. Writeaprogramtoimplement Bresenham'slinedrawingalgorithm.
2. Writeaprogramtoimplementmid-pointcircledrawingalgorithm.
3. WriteaprogramtoclipalineusingCohenandSutherlandlineclippingalgorithm.
4. WriteaprogramtoclipapolygonusingSutherlandHodgemanalgorithm.
5. Writeaprogramto applyvarious2Dtransformationsona2D object (usehomogenouscoordinates).
6. Writeaprogramtoapplyvarious3Dtransformationsona3Dobject andthenapply parallel andperspectiveprojectionon it
7. WriteaprogramtodrawHermite/Beziercurve