
**PUNE VIDYARTHI GRIHA'S**

**COLLEGE OF ENGINEERING AND TECHNOLOGY, PUNE-9**

**(AFFILIATED TO UNIVERSITY OF PUNE, PUNE)**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CURRICULUM BOOK**

**ACADEMIC YEAR: 2019-20**

**FOR THE PROGRAMME**

**S. E. (COMPUTER ENGINEERING)**

PUNE VIDYARTHI GRIHA'S
COLLEGE OF ENGINEERING & TECHNOLOGY, PUNE-9

# Curriculum book of SE (COMPUTER)

2019-20


**PUNE VIDYARTHI GRIHA'S**

**COLLEGE OF ENGINEERING AND TECHNOLOGY**

## VISION

**TO ACHIEVE EXCELLENCE IN ENGINEERING EDUCATION**

## MISSION

- **To satisfy all stakeholders**
- **To develop ethical, highly motivated engineering professionals with good human values, requisite skills and competencies**
- **To adopt innovative teaching mechanisms**
- **To promote research culture**
- **To contribute to country's economic development**
- **To be responsive to changes in technology, socio-economic and environmental conditions**

PUNE VIDYARTHI GRIHA'S
COLLEGE OF ENGINEERING & TECHNOLOGY, PUNE-9

# Curriculum book of SE (COMPUTER)

2019-20

## Department of Computer Engineering

**Vision of the Department**

**To empower students to meet challenges of 21st century by imparting quality education in Computer Engineering**

**Mission**

1) **To impart knowledge through innovative teaching-learning process to cater the needs of industries and higher education**
2) **To inculcate good human values, professional competencies and create awareness about global technologies in the field of Computer Engineering**

**PEOs**

**The students of Computer engineering after passing out should**

**PEO1.** Graduates of the program will possess strong fundamental concepts in engineering science and Technology to address future challenges of Computer Engineering.

**PEO2.** Possess knowledge and skills in the field of Computer Science and Engineering for solving engineering problems with innovative approaches

**PEO3.** Possess behavioral aspects for research, entrepreneurship and higher studies in the field of Computer Science & Information Technology.

**PEO4.** Have commitment to ethical practices, societal contributions through communities and life-long learning**.**

**PEO5.** Possess better interpersonal and presentation skills leading to responsible & competent professionals at global level.

**POs:**

The students in the CSE program will attain: (Course Objectives)

1. an ability to apply knowledge of computing, mathematics including discrete mathematics as well as probability and statistics, science, and engineering and technology;
2. an ability to identify a problem and provide a systematic solution by conducting experiments, as well as analyzing and interpreting the data;
3. an ability to design, implement, and evaluate a software or a software/hardware system, component, or process to meet desired needs within realistic constraints;
4. an ability to identify, formulate, and provide systematic solutions to complex engineering problems and validate the solution;
5. an ability to use the techniques, skills, modern engineering tools and technologies necessary for practice as a IT professional;
6. an ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems with necessary constraints and assumptions;
7. an ability to analyze the local and global impact of computing on individuals, organizations and society;
8. an ability to understand professional, ethical, legal, security and social issues and responsibilities;
9. an ability to function effectively as an individual or as a team member to accomplish a desired goal(s) in multidisciplinary environment;
10. an ability to engage in life-long learning and continuing professional development to cope up with fast changes in the technologies/tools with the help of open electives, professional organizations and extra-curricular activities;
11. an ability to communicate effectively in engineering community at large by means of effective presentations, report writing, paper publications, demonstrations;
12. an ability to understand engineering, management, financial aspects, performance, optimizations and time complexity necessary for professional practice;
13. an ability to apply design and development principles in the construction of software systems of varying complexity.

# INDEX

# SE COMPUTER

## Savitribai Phule Pune University

## Second Year of Computer Engineering (2015 Course)

(With effect from Academic Year 2016-17)

### Semester I

| Course Code | Course Name | Teaching Scheme Hours / Week Theory | Tutorial | Practical | Examination Scheme & Marks In-Sem | End-Sem | TW | PR | OR | Total | Credit TH + TUT | PR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 210241 | Discrete Mathematics | 04 | -- | -- | 50 | 50 | -- | -- | -- | 100 | 04 | -- |
| 210242 | Digital Electronics and Logic Design | 04 | -- | -- | 50 | 50 | -- | -- | -- | 100 | 04 | -- |
| 210243 | Data Structures and Algorithms | 04 | -- | -- | 50 | 50 | -- | -- | -- | 100 | 04 | -- |
| 210244 | Computer Organization and Architecture | 04 | -- | -- | 50 | 50 | -- | -- | -- | 100 | 04 | -- |
| 210245 | Object Oriented Programming | 04 | -- | -- | 50 | 50 | -- | -- | -- | 100 | 04 | -- |
| 210246 | Digital Electronics Lab | -- | -- | 02 | -- | -- | 25 | 50 | -- | 75 | -- | 01 |
| 210247 | Data Structures Lab | -- | -- | 04 | -- | -- | 25 | 50 | -- | 75 | -- | 02 |
| 210248 | Object Oriented Programming Lab | -- | -- | 02 | -- | -- | 25 | 50 | -- | 75 | -- | 01 |
| 210249 | Soft Skills | -- | -- | 02 | -- | -- | 25 | -- | -- | 25 | -- | 01 |
| | | | **Total** | | | | | | | | 20 | 05 |
| 210250 | Audit Course 1 | -- | -- | -- | -- | -- | -- | -- | -- | -- | Grade | |
| | **Total** | 20 | -- | 10 | 250 | 250 | 100 | 150 | -- | 750 | | 25 |

**Abbreviations:**

| | | | |
|---|---|---|---|
| TW: | Term Work | TH: | Theory |
| OR: | Oral | TUT: | Tutorial |
| PR: | Practical | Sem: | Semester |

# SE COMPUTER

# Semester-I

## Discrete Mathematics

<table>
<tr><td colspan="2">Course Title: Discrete Mathematics</td><td colspan="2">Course Number: 210241</td><td>Course Code:</td></tr>
<tr><td colspan="2">Year: 2019-20</td><td colspan="3">Semester: I</td></tr>
<tr><td colspan="2">Designation of Course</td><td colspan="3">Professional Core</td></tr>
<tr><td colspan="2">Teaching Scheme: 4 Hrs/Week</td><td colspan="3">Tutorial: -</td></tr>
<tr><td rowspan="3">Course Assessment Methods</td><td rowspan="2">Direct methods</td><td>On-line/In-semester Examination: 50 Marks</td><td colspan="2">End Semester Examination: 50 Marks</td></tr>
<tr><td>Unit Test</td><td colspan="2">Practical/Oral/Term Work</td></tr>
<tr><td>Indirect Methods</td><td colspan="3">Assignments, Presentations<br>Quiz, Q&A session, Group Discussion</td></tr>
<tr><td>Prerequisites</td><td colspan="4">Basic Mathematics</td></tr>
<tr><td colspan="5">Course Objectives</td></tr>
<tr><td>1</td><td colspan="4">• To use appropriate set and Propositional Logic to analyze practical examples, interpret the associated operations and terminology in context.</td></tr>
<tr><td>2</td><td colspan="4">• Determine number of logical possibilities of events.</td></tr>
<tr><td>3</td><td colspan="4">• Learn logic and proof techniques to expand mathematical maturity.</td></tr>
<tr><td>4</td><td colspan="4">• Formulate problems precisely, solve the problems, apply formal proof techniques, and explain the reasoning clearly.</td></tr>
<tr><td colspan="5">Course Outcomes</td></tr>
<tr><td>CO1</td><td colspan="4">• Solve real world problems logically using appropriate set, function, and relation models and interpret the associated operations and terminologies in context.</td></tr>
<tr><td>CO2</td><td colspan="4">• Analyze and synthesize the real world problems using discrete mathematics.</td></tr>
</table>

| Course Contents | |
|---|---|
| **Unit-I** | **Set Theory and Logic** |
| | Discrete Mathematics, Significance of Discrete Mathematics in Computer Engineering,<br>**Sets–**<br>Naïve Set Theory (Cantorian Set Theory), Axiomatic Set Theory, Need for Sets, Representation of Sets, Set Operations, cardinality of set , **Types of Sets** – Bounded and Unbounded Sets, Countable and Uncountable Sets, Finite and Infinite Sets, Countably Infinite and Uncountably Infinite Sets, Countability of Rational Numbers Using Cantor Diagonalization Argument, power set,<br>**Propositional Logic-** logic**,** Propositional Equivalences, Application of Propositional LogicTranslating English Sentences, Proof by Mathematical Induction and Strong Mathematical Induction. |
| | **Practical/Tutorial** |
| | - |
| **Unit-II** | **Relations and Functions** |
| | Relations and Their Properties, n-ary Relations and Their Applications, Representing Relations , Closures of Relations, Equivalence Relations, Partial Orderings, partitions, Hasse Diagram, Lattices, Chains and Anti-Chains, Transitive Closure and Warshall's Algorithm, n-Ary Relations and their Applications.<br>**Functions-** Surjective, Injective and Bijective functions, Inverse Functions and Compositions of Functions, The Pigeonhole Principle. |
| | **Practical/Tutorial** |
| | - |
| **Unit-III** | **Counting** |

| | |
|---|---|
| | The Basics of Counting, rule of Sum and Product, Permutations and Combinations, Binomial Coefficients and Identities, Generalized Permutations and Combinations, Algorithms for generating Permutations and Combinations. |
| | **Practical/Tutorial** |
| | - |
| **Unit-IV** | **Graph Theory** |
| | Graphs and Graph Models, Graph Terminology and Special Types of Graphs, Representing Graphs and Graph Isomorphism, Connectivity, Euler and Hamilton Paths, Single source shortest path-Dikjtra's Algorithm, Planar Graphs, Graph Colouring. Case Study- Web Graph, Google map |
| | **Practical/Tutorial** |
| | |
| **Unit- V** | **Trees** |
| | Introduction, properties of trees, Binary search tree, decision tree, prefix codes and Huffman coding, cut sets, Spanning Trees and Minimum Spanning- Kruskal's and Prim's algorithms, The Max flow- Min Cut Theorem (Transport network). Case Study- Game Tree, Min-Max Tree. |
| | **Practical/Tutorial** |
| | - |
| **Unit-VI** | **Algebraic Structures and Coding Theory** |
| | The structure of algebra, Algebraic Systems, Semi Groups, Monoids, Groups, Homomorphism and Normal Subgroups, and congruence relations, Rings, Integral Domains and Fields, coding theory, Polynomial Rings and polynomial Codes, Galois Theory –Field Theory and Group Theory. |
| | **Practical/Tutorial** |
| | - |

| Text Books | Author | Title of Book | Publication |
|---|---|---|---|
| T1 | Kenneth H. Rosen | Discrete Mathematics and its Application | Tata McGraw-Hill |
| T2 | C.L.Liu | Elements of Discrete Mathematics | Tata McGraw-Hill |
| **Reference Books** | | | |
| R1 | B. Kolman, R. Busby and S. Ross | Discrete Mathematical Structures | Pearson Education |
| R2 | N.Biggs | Discrete Mathematics | Oxford University Press |
| R3 | Narsingh Deo | Graph with application to engineering and Computer Science | Prentice Hall of India |
| R4 | Dr.K.D.Joshi | Foundation of Discrete Mathematics | New Age International Ltd |
| R5 | Eric Gossett | Discrete Mathematical Structures with Proofs‖, | Wiley India Ltd, |
| R6 | Sriram P. and Steven S. | Computational Discrete Mathematics | Cambridge University Press |
| **Self-Learning Facilities, Web Resources,** | NPTEL videos | | |

| | |
|---|---|
| **Research papers for reference** | |
| **Contents beyond Syllabus** | Probability |
| **Additional Experiments** | - |
| **Bridging Courses** | NPTEL Online course introduction to students for D.M. |
| **Tutorials/Assignments** | Assignments 1:<br>Assignments 2: |
| **Presentations** | Graph Coloring<br>Google Map<br>Etc… |

**Digital Electronics and Logic Design**

<table>
<tr><td colspan="2">Course Title:<br>Digital Electronics & Logic Design</td><td colspan="2">Course Number:</td><td>Course Code:<br>210242</td></tr>
<tr><td colspan="2">Year: 2019-20</td><td colspan="3">Semester: I</td></tr>
<tr><td colspan="2">Designation of Course</td><td colspan="3">Professional Core</td></tr>
<tr><td colspan="2">Teaching Scheme: 4 Hrs/Week</td><td colspan="3">Tutorial:</td></tr>
<tr><td rowspan="3">Course Assessment Methods</td><td rowspan="2">Direct methods</td><td>On-line Examination: 50 Marks</td><td colspan="2">End Semester Examination: 50 Marks</td></tr>
<tr><td></td><td colspan="2">Practical/Oral/Term Work</td></tr>
<tr><td>Indirect Methods</td><td>Assignments, Presentations</td><td colspan="2">Seminars, Quiz, Q&A session,<br>Group Discussion</td></tr>
<tr><td>Prerequisites</td><td colspan="4">Basic Electronics Engineering</td></tr>
<tr><td colspan="5">Course Objectives</td></tr>
<tr><td>1</td><td colspan="4">To understand the functionality and design of Combinational and Sequential Circuits</td></tr>
<tr><td>2</td><td colspan="4">To understand and compare the functionalities, properties and applicability of Logic Families.</td></tr>
<tr><td>3</td><td colspan="4">To understand concept of programmable logic devices and ASM chart and get acquainted with design of synchronous state machines.</td></tr>
<tr><td>4</td><td colspan="4">To design and implement digital circuits using VHDL.</td></tr>
<tr><td>5</td><td colspan="4">To understand the functionality of microcontroller.</td></tr>
<tr><td colspan="5">Course Outcomes<br>On completion of the course, student will be able to–</td></tr>
<tr><td>CO1</td><td colspan="4"></td></tr>
</table>

| | |
|---|---|
| | Realize and simplify Boolean Algebraic assignments for designing digital circuits using K-Maps. |
| CO2 | Design and implement Sequential and Combinational digital circuits as per the specifications. |
| CO3 | Apply the knowledge to select the logic families IC packages as per the design specifications. |
| CO4 | Design the minimum systems using VHDL. |
| CO5 | Develop minimum embedded system for simple real world application. |
| **Course Contents** | |
| **Unit-I** | **Combinational Logic Design** |
| | **Logic minimization:** Representation of truth-table, Sum of Product (SOP) form, Product of Sum (POS) form, Simplification of logical functions, Minimization of SOP and POS forms using K-Maps up to 4 variables and **Quine-** McCluskey Technique, realization of logic gates. **Design of Combinational Logic:** Code converter - BCD, Excess-3, Gray code, Binary Code. Half- Adder, Full Adder, Half Subtractor, Full Subtractor, Binary Adder (IC 7483), BCD adder, Look ahead carry generator, Multiplexers (MUX): MUX (IC 74153, 74151), MUX tree, Demultiplexers (DEMUX)- Decoder. (IC 74138, IC 74154). DMUX Tree, Implementation of SOP and POS using MUX, DMUX, Comparators, Parity generators and Checker, Priority Encoders. |
| | **Practical/Tutorial** |
| | • To Realize Full Adder and Subtractor using a) Basic Gates and b) Universal Gates<br>• Design and implement Code converters-Binary to Gray and BCD to Excess-3<br>• Design of n-bit Carry Save Adder (CSA) and Carry Propagation Adder (CPA). Design and Realization of BCD Adder using 4-bit Binary Adder (IC 7483).<br>• Realization of Boolean Expression for suitable combination logic using MUX 74151 / DMUX 74154<br>• To Verify the truth table of one bit and two bit comparators |

| | |
|---|---|
| | using logic gates and comparator IC<br>• Design & Implement Parity Generator using EX-OR. |
| **Unit-II** | **Sequential Logic Design** |
| | **Flip- flop:** SR, JK, D, T; Preset & Clear, Master and Slave Flip Flops, Truth Tables and Excitation tables, Conversion from one type to another type of Flip Flop. **Registers:** Buffer register, shift register, Applications of shift registers. **Counters:** Asynchronous counter. Synchronous counter, ring counters, BCD Counter, Johnson Counter, Modulus of the counter (IC 7490). **Synchronous Sequential Circuit Design:** Models – Moore and Mealy, State diagram and State Tables, Design Procedure, Sequence generator and detector. **Asynchronous Sequential Circuit Design:** Difference with synchronous circuit design, design principles and procedure, applications. |
| | **Practical/Tutorial** |
| | • Filp Flop Conversion: Design and Realization<br>• Design and implement a system using flip-flops, to monitor number of vehicles entering and exiting from a car parking area with maximum capacity of 15 and having separate entry and exit gates.<br>• Design of Ripple Counter using suitable Flip Flops<br>• Realization of 3 bit Up/Down Counter using MS JK Flip Flop / D Flip Flop<br>• b. Realization of Mod -N counter using ( 7490 and 74193 )<br>• Assume a scenario of a hall where students are entering to attend seminar. Design and implement a system which will increment count if student is entering in the hall and will decrement count if student is exiting the hall. Assume seating capacity of a hall is 63.<br>• Design and Realization of Ring Counter and Johnson Ring counter<br>• Design and implement Sequence generator using JK flip-flop.<br>• Design and implement Sequence detector using JK flip-flop.<br>• Design and implement pseudo random sequence generator.<br>• Study of Shift Registers ( SISO,SIPO, PISO,PIPO ) |
| **Unit-III** | **Algorithmic State Machines** |
| | **Algorithmic State Machines:** Finite State Machines (FSM) and ASM, ASM charts, notations, construction of ASM chart and realization for |

| | |
|---|---|
| | sequential circuits, Sequence Generator, Types of Counters. **VHDL:** Introduction to HDL, Data Objects & Data Types, Attributes., VHDL-Library, Design Entity, Architecture, Modeling Styles, Concurrent and Sequential Statements,<br>**Design Examples:** VHDL for Combinational Circuits-Adder, MUX, VHDL for Sequential Circuits, Synchronous and Asynchronous Counter. |
| | **Practical/Tutorial** |
| | • Design of ASM chart using MUX controller Method<br>• Design and simulation of - Full adder , Flip flop, MUX using VHDL (Any 2)<br>Use different modeling styles.<br>• Design & simulate asynchronous 3- bit counter using VHDL. |
| **Unit-IV** | **Programmable Logic Devices** |
| | ROM as PLD, Programmable Logic Array (PLA), Programmable Array Logic (PAL), Designing combinational circuits using PLDs. |
| | **Practical/Tutorial** |
| | • Design and Implementation of Combinational Logic using PLAs.<br>• Design and Implementation of Combinational Logic using PLAs. |
| **Unit- V** | **Logic Families** |
| | **Classification of logic families:** Unipolar and Bipolar Logic Families, Characteristics of Digital ICs: Speed, power dissipation, figure of merits, fan-out, Current and voltage parameters, Noise immunity, operating temperature range, power supply requirements. **Transistor-Transistor Logic:** Operation of TTL, Current sink logic, TTL with active pull up, TTL with open collector output, Schottkey TTL, TTL characteristics, TTL 5400/7400 series, **CMOS:** CMOS Inverter, CMOS characteristics, CMOS configurations- Wired Logic, Open drain outputs, **Interfacing:** TTL to CMOS and CMOS to TTL. Tristate Logic and Tristate TTL inverter. |
| | **Practical/Tutorial** |
| | • Study of TTL Logic Family: Feature, Characteristics and Comparison with CMOS Family |
| **Unit-VI** | **Microcontrollers** |

| | | | |
|---|---|---|---|
| | Comparison of typical microprocessor and microcontroller. **Microcontroller 8051:** Features, architecture, Pin description, **Programming model**– Special Function Registers, addressing modes, instruction set, Timers and Counters, serial communication, interrupts, interfacing with ADC and DAC | | |
| | **Practical/Tutorial** | | |
| | • Study of Microcontroller 8051 : Features, Architecture and Programming Model | | |
| **Text Books** | **Author** | **Title of Book** | **Publication** |
| T1 | R.P. Jain | Modern Digital Electronics | TMH, 2012, ISBN–13: 978-0-07- 066911-6 |
| T2 | Stephen Brown, Zvonko Vranesic | Fundamentals of Digital Logic with VHDL Design | McGraw-Hill, ISBN–13:978-1-25-902597-6. |
| T3 | Muhammas Mazidi, Janice Mazidi and Rolin McKinlay | The 8051 Microcontroller and<br><br>Embedded Systems using Assembly and C | Pearson Education, ISBN-13: 9788131758991 |
| **Reference Books** | | | |
| R1 | John Yarbrough | Digital Logic applications and Design | Cengage Learning |
| R2 | D. Leach, Malvino, Saha | Digital Principles and Applications | Tata McGraw Hill |
| R3 | Anil Maini | Digital Electronics: Principles and Integrated Circuits | Wiley India Ltd |
| R4 | | | |

| | | | |
|---|---|---|---|
| | Norman B & Bradley | Digital Logic Design Principles | Wiley India Ltd |
| R5 | Scott Mackenzie | The 8051 Microcontroller | Prentice Hall India |
| **Self-Learning Facilities, Web Resources, Research papers for reference** | Nil | | |
| **Contents beyond Syllabus** | | | |
| **Additional Experiments** | | | |
| **Bridging Courses** | | | |
| **Tutorials** | | | |
| **Presentations** | | | |

## Discrete Structures & Algorithms

| | | | |
|---|---|---|---|
| **Course Title: DSA** | | **Course Number:** | **Course Code:210243** |
| **Year: 2019-20** | | **Semester: I** | |
| **Designation of Course** | | Professional Core/Elective/Humanities | |
| **Teaching Scheme:** 4 Hrs/Week | | **Tutorial:** | |
| **Course Assessment Methods** | **Direct methods** | On-line/In-semester Examination: 50/30 Marks | End Semester Examination: 50/70 Marks |
| | | | Practical/Oral/Term Work |
| | **Indirect Methods** | Assignments, Presentations | Seminars, Quiz, Q&A session, Group Discussion |
| **Prerequisites** | | | |
| **Course Objectives** | | | |
| 1 | To understand the standard and abstract data representation methods. | | |
| 2 | To acquaint with the structural constraints and advantages in usage of the data. | | |
| 3 | To understand the memory requirement for various data structures. | | |
| 4 | To operate on the various structured data. | | |
| 5 | To understand various data searching and sorting methods with pros and cons. | | |
| 6 | To understand various algorithmic strategies to approach the problem solution. | | |
| **Course Outcomes** | | | |
| 1 | To discriminate the usage of various structures in approaching the problem solution. | | |
| 2 | To design the algorithms to solve the programming problems. | | |
| 3 | To use effective and efficient data structures in solving various Computer Engineering domain problems. | | |
| 4 | To analyze the problems to apply suitable algorithm and data structure. | | |
| 5 | To use appropriate algorithmic strategy for better efficiency | | |
| **Course Contents** | | | |
| **Unit-I** | **Introduction to Algorithm and Data Structures :** | | |
| | **Algorithms-** Problem Solving, Introduction to Algorithms, Characteristics of algorithms, Algorithm design tools: Pseudo code and flowchart, Analysis of Algorithms, Complexity of algorithms- Space complexity, Time complexity, Asymptotic notation- Big-O, Theta and Omega, standard measures of efficiency.<br>**Data Structures-** Data structure, Abstract Data Types (ADT), Concept of linear and Non-linear, static and dynamic, persistent and ephemeral data structures, and relationship among data, data structure, and algorithm, From Problem to Program.<br>**Algorithmic Strategies-** Introduction to algorithm design strategies- Divide | | |

| | |
|---|---|
| | and Conquer, and Greedy strategy.<br>**Recurrence relation** - Recurrence Relation, Linear Recurrence Relations, With constant Coefficients, Homogeneous Solutions. Solving recurrence relations |
| | |
| | **Practical/Tutorial** |
| | |
| **Unit-II** | **Linear Data Structures Using Sequential Organization** |
| | Sequential Organization, Linear Data Structure Using Sequential Organization, Array as an Abstract Data Type, Memory Representation and Address Calculation, Inserting an element into an array, Deleting an element, Multidimensional Arrays, Two-dimensional arrays, n- dimensional arrays, Concept of Ordered List, Single Variable Polynomial, Representation using arrays, Polynomial as array of structure, Polynomial addition, Polynomial multiplication, Sparse Matrix, Sparse matrix representation, Sparse matrix addition, Transpose of sparse matrix, String Manipulation Using Array.<br>**Case Study-** Use of sparse matrix in Social Networks and Maps. |
| | **Practical/Tutorial** |
| | 1. Write C/C++ program to store marks scored for first test of subject 'Data Structures and Algorithms' for N students. Compute<br>i. The average score of class<br>ii. Highest score and lowest score of class<br>iii. Marks scored by most of the students<br>iv. list of students who were absent for the test<br>2. Write C/C++ program for storing matrix. Write functions for<br>i. Check whether given matrix is upper triangular or not<br>ii. Compute summation of diagonal elements<br>iii. Compute transpose of matrix<br>iv. Add, subtract and multiply two matrices<br>Write C++ program with class for String. Write a function<br>i. *frequency* that determines the frequency of occurrence of particular character in the string.<br>ii. *delete* that accepts two integers, start and length. The function computes a new string that is equivalent to the original string, except that length characters being at start have been removed.<br>iii. *chardelete* that accepts a character c. The function returns the string with all occurrences of c removed.<br>iv. *replace* to make an in - place replacement of a substring w of a string by the string x. note that w may not be of same size of x<br>v. *palindrome* to check whether given string is palindrome or not |
| **Unit-III** | **Linked Lists** |
| | Concept, Comparison of sequential and linked organizations, Primitive operations, Realization of Linked Lists, Realization of linked list using arrays, Dynamic Memory Management, Linked list using dynamic memory management, Linked List Abstract Data Type, Linked list operations, Head pointer and header node,<br>**Types of linked list-** Linear and circular linked lists, Doubly Linked List and operations, Circular Linked List, Singly circular linked list, Doubly circular |

| | |
|---|---|
| | linked list**, Polynomial Manipulations -** Polynomial addition, Multiplication of two polynomials using linked list.<br>**Generalized Linked List (GLL)** concept, representation of polynomial and sets using GLL.<br>**Case Study-** Garbage Collection |
| | **Practical/Tutorial** |
| | 1. Write C++ program to store set of negative and positive numbers using linked list. Write functions to<br>i. Insert numbers<br>ii. Delete nodes with negative numbers<br>iii. Create two more linked lists using this list, one containing all positive numbers and other containing negative numbers<br>For two lists that are sorted; Merge these two lists into third resultant list that is sorted.<br>2. Write C++ program for storing binary number using doubly linked lists. Write functions-<br>i. to compute 1's and 2's complement ii. add two binary numbers<br><br>3. Let x = ($x_1$,$x_2$, ... ,$x_n$) and y = ($y_1$, $y_2$,.... , $y_m$) be two Circular linked lists. Assume that in each linked list, the nodes are in non-decreasing order of their data-field values. Write C/C++ program to merge the two lists to obtain a new linked list z in which the nodes are also in this order. Following the merge, x and y should represent empty lists because each node initially in x or y is now in z. No additional nodes may be used |
| **Unit-IV** | **Stacks** |
| | **Stacks-** concept, Primitive operations, Stack Abstract Data Type, Representation of Stacks Using Sequential Organization, stack operations, Multiple Stacks, Applications of Stack- Expression Evaluation and Conversion, Polish notation and expression conversion, Need for prefix and postfix expressions, Postfix expression evaluation, Linked Stack and Operations<br>**Recursion-** concept, **variants of recursion-** direct, indirect, tail and tree, Backtracking algorithmic strategy, use of stack in backtracking.<br>**Case Study-** 4 queens problem, Android- multiple tasks/multiple activities and back stack. |
| | **Practical/Tutorial** |
| | 1. A palindrome is a string of character that's the same forward and backward. Typically, punctuation, capitalization, and spaces are ignored. For example, "Poor Dan is in a droop" is a palindrome, as can be seen by examining the characters "poor dan is in a droop" and observing that they are the same forward and backward. One way to check for a palindrome is to reverse the characters in the string and then compare with them the original-in a palindrome, the sequence will be identical. Write C++ program with functions-<br>i. To check whether given string is palindrome or not that uses a |

| | |
|---|---|
| | stack to determine whether a string is a palindrome.<br>ii. to remove spaces and punctuation in string, convert all the Characters to lowercase, and then call above Palindrome checking function to check for a palindrome<br>iii. to print string in reverse order using stack<br>2. Implement C++ program for expression conversion as infix to postfix and its evaluation using stack based on given conditions<br>i. Operands and operator, both must be single character.<br>ii. Input Postfix expression must be in a desired format.<br>Only '+', '-', '*' and '/ ' operators are expected |
| **Unit- V** | **Queues** |
| | Concept, Queue as Abstract Data Type, Realization of Queues Using Arrays , Circular Queue, Advantages of using circular queues, Multi-queues, Deque, Priority Queue, Array implementation of priority queue, Linked Queue and operations.<br>**Case study- P**riority queue in bandwidth management |
| | **Practical/Tutorial** |
| | 1. Queues are frequently used in computer programming, and a typical example is the creation of a job queue by an operating system. If the operating system does not use priorities, then the jobs are processed in the order they enter the system. Write C++ program for simulating job queue. Write functions to add job and delete job from queue.<br>2. A double-ended queue (deque) is a linear list in which additions and deletions may be made at either end. Obtain a data representation mapping a deque into a one-dimensional array. Write C++ program to simulate deque with functions to add and delete elements from either end of the deque |
| **Unit-VI** | **Sorting and Searching** |
| | **Searching-** Search Techniques, Sequential search, variant of sequential search- sentinel search, Binary search, Fibonacci search.<br>**Case Study**- Use of Fibonacci search in non-uniform access memory storage and in Optimization of Unimodal Functions.<br>Sorting- Types of sorting-Internal and external sorting, General sort concepts- sort order, stability, efficiency, number of passes, Sorting methods- Bubble sort, Insertion sort, Selection sort, Quick sort, Heap sort, Shell sort, Bucket sort, Radix sort, Comparison of All Sorting Methods.<br>**Case Study-** Timsort is a hybrid stable sorting algorithm. |
| | **Practical/Tutorial** |
| | 1. Write C++ program to store roll numbers of student in array who attended training program in random order. Write function for-<br>i. Searching whether particular student attended training program or not using linear search and sentinel search.<br>ii. Searching whether particular student attended training program or not using binary search and Fibonacci search.<br>2. Write C++ program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using i)Selection Sort ii) Bubble sort and display top five scores<br>3. Write C++ program to store second year percentage of students in array. |

| | | | |
|---|---|---|---|
| | Write function for sorting array of floating point numbers in ascending order using i)Insertion sort ii) Shell Sort and display top five scores<br>4. Write C++ program to store first year percentage of students in array. Sort array of floating point numbers in ascending order using quick sort and display top five scores. | | |
| **Text Books** | **Author** | **Title of Book** | **Publication** |
| T1 | Brassard & Bratley | Fundamentals of Algorithmics | Prentice Hall India/Pearson Education, ISBN 13-9788120311312 |
| T2 | Horowitz and Sahani | Fundamentals of Data Structures in C++ | University Press, ISBN 10: 0716782928 ISBN 13: 9780716782926 |
| T3 | Goodrich, Tamassia, Goldwasser | Data Structures and Algorithms in C++ | Wiley publication, ISBN-978-81-265-1260-7 |
| **Reference Books** | | | |
| R1 | R. Gillberg, B. Forouzn | Data Structures: A Pseudo code approach with C | Cenage Learning, ISBN 9788131503140 |
| R2 | Horowitz, Sahani and Rajshekaran | Fundamentals of Computer Algorithms | University Press, ISBN-13, 9788175152571 |
| R3 | Yedidyah Langsam, Moshe J Augenstein, Aron M Tenenbaum | Data Structures using C and C++ | Pearson Education, ISBN 81-317-0328-2 |
| R4 | A Michael Berman | Data Structures via C++: Objects by Evolution | Oxford University Press, ISBN:0-19-510843-4 |
| R5 | M. Weiss | Data Structures and Algorithm Analysis in C++ | 2nd edition, Pearson Education, 2002, ISBN-81-7808-670-0 |
| **Self-Learning Facilities, Web Resources, Research papers for reference** | NPTEL | | |
| **Contents beyond Syllabus** | Core dumped error(Segmentation Fault) | | |
| **Additional Experiments** | - | | |

| | |
|---|---|
| **Bridging Courses** | - |
| **Tutorials** | - |
| **Presentations** | - |

**Computer Organization & Architecture**

| | | | |
|---|---|---|---|
| **Course Title:** Computer Organization and Architecture | | **Course Number:** 210244 | **Course Code:** 210244 |
| **Year:** 2019-20 | | **Semester:** I | |
| **Designation of Course** | | Professional Core | |
| **Teaching Scheme:** 4 Hrs/Week | | **Tutorial:** | |
| **Course Assessment Methods** | **Direct methods** | On-line Examination: 50 Marks | End Semester Examination: 50 Marks |
| | | | Practical/Oral/Term Work |
| | **Indirect Methods** | Assignments, Objective test | Quiz, Q&A session, Group Discussion |
| **Prerequisites** | • Fundamentals of Programming Languages-I & II and<br>• Basics of Electronics Engineering | | |
| **Course Objectives** | | | |
| 1 | To understand the structure, function and characteristics of computer systems | | |
| 2 | To understand the design of the various functional units and components of digital computers | | |
| 3 | To identify the elements of modern instructions sets and explain their impact on processor design. | | |
| 4 | To explain the function of each element of a memory hierarchy, identify and compare different methods for computer I/O | | |
| 5 | To compare simple computer architectures and organizations based on established performance metrics. | | |
| **Course Outcomes** | | | |
| CO1 | Demonstrate computer architecture concepts related to design of modern processors, memories and I/Os. | | |
| CO2 | Analyze the principles of computer architecture using examples drawn from commercially available computers. | | |
| CO3 | Evaluate various design alternatives in processor organization. | | |
| **Course Contents** | | | |
| **Unit-I** | **Computer Evolution and Performance** | | |
| | Computer Organization and Architecture, Structure and Function, Evolution (a brief history) of computers, Designing for Performance, Evolution of Intel processor architecture- 4 bit to 64 bit, performance assessment.<br>**A top level view of Computer function and interconnection-** Computer Components, Computer Function, Interconnection structure, bus interconnection,<br>**Computer Arithmetic-** The Arithmetic and Logic Unit, addition and subtraction of signed numbers, design of adder and fast adder, carry look ahead addition, multiplication of positive numbers, signed operand multiplication, booths algorithm, fast multiplication, integer division.<br>**Floating point representation and operations –** IEEE standard, arithmetic operations, guard bits and truncation. | | |
| | **Practical/Tutorial** | | |

| | |
|---|---|
| | |
| **Unit-II** | **Computer Memory System** |
| | Characteristics of memory system, The memory hierarchy. **Cache Memory-** Cache memory principles, Elements of cache design- cache address, size, mapping functions, replacement algorithms, write policy, line size, number of cache, one level and two level cache, performance characteristics of two level cache- locality & operations. **Case Study-** pentium4 cache organization. **Internal Memory-** semiconductor main memory, advanced DRAM organization. **External Memory-** Hard Disk organization, RAID- level 1 to level 6. |
| | **Practical/Tutorial** |
| | |
| **Unit-III** | **Input and Output System** |
| | External devices, **I/O modules-** Module function and I/O module structure, **Programmed I/O-** overview, I/O commands, I/O instructions, Interrupt driven I/O- interrupt processing, design issues. **Case Study-** Study of Programmable Interrupt Controller 82C59A in brief. **Direct Memory Access-** drawbacks of programmed and interrupt driven I/O, DMA functions, **Case Study- DMA Controller** Intel 8237A-study in brief, I/O channels and processors- evolution and characteristics, The external Interface- Thunderbolt and Infinite Band |
| | **Practical/Tutorial** |
| | |
| **Unit-IV** | **Instruction Sets** |
| | **Characteristics and Functions-** machine instruction characteristics, types of operands, **Case Study-**Intel 8086,<br>**Types of operations-** data transfer, arithmetic, logical, conversion, input-output, system control, and transfer of control,<br>**Case Study-**Intel 8086 operation types. **Addressing modes and Formats-** Addressing modes- immediate, direct, indirect, register, register indirect, displacement and stack, **Case Study-**8086 addressing modes, **Instruction Formats-** instruction length, allocation of bits, variable length instructions. **Case Study-** 8086 instruction formats. |
| | **Practical/Tutorial** |
| | |
| **Unit- V** | **Processor Organization** |
| | Processor organization, Register organization- user visible registers, control and status registers, **Case Study-** register organization of microprocessor 8086.<br>**Instruction Cycle-** The indirect cycle and Data flow.<br>**Instruction Pipelining-** Pipelining Strategy, pipeline performance, pipeline hazards, dealing with branches, **Case Study-** Intel 586 pipelining.<br>**Instruction level parallelism and superscalar processors-** Super scalar verses super pipelined, constraints,<br>**Design Issues-** instruction level and machine parallelism, Instruction issue policy, register renaming, machine parallelism, branch prediction, superscalar execution and implementation. **Case study-** Pentium 4. |

| | | | |
|---|---|---|---|
| | **Practical/Tutorial** | | |
| | | | |
| **Unit-VI** | **Basic Processing Unit** | | |
| | **Fundamental Concepts-** register transfer, performing arithmetic or logic operations, fetching a word from memory, storing a word in memory, Execution of a complete instruction- branch instructions,<br>**Hardwired control**, M**icro-programmed control-** micro instructions, micro program sequencing, wide branch addressing, microinstruction with next address field, prefetching microinstructions and emulation. | | |
| | **Practical/Tutorial** | | |
| | | | |
| **Text Books** | **Author** | **Title of Book** | **Publication** |
| T1 | W. Stallings | Computer Organization and Architecture: Designing for performance | Pearson Education/ Prentice Hall of India |
| T2 | Zaky S, Hamacher | Computer Organization | McGraw-Hill Publications |
| **Reference Books** | | | |
| R1 | John P Hays | Computer Architecture and Organization | McGraw-Hill Publication |
| R2 | Miles Murdocca and Vincent Heuring | Computer Architecture and Organization- an integrated approach | Wiley India Pvt. Ltd |
| R3 | A. Tanenbaum | Structured Computer Organization | Prentice Hall of India |
| R4 | Patterson and Hennessy | Computer Organization and Design | Morgan Kaufmann Publishers In |
| **Self-Learning Facilities, Web Resources, Research papers for reference** | Nil | | |
| **Contents beyond Syllabus** | | | |
| **Additional Experiments** | | | |
| **Bridging Courses** | | | |
| **Tutorials** | | | |
| **Presentations** | | | |

**Object Oriented Programming**

| | | | |
|---|---|---|---|
| **Course Title: OOP** | | **Course Number:** | **Course Code: 210245** |
| **Year: 2019-20** | | **Semester: II** | |
| **Designation of Course** | | Professional Core/Elective/Humanities | |
| **Teaching Scheme:** 3 Hrs/Week | | **Tutorial:** | |
| **Course Assessment Methods** | **Direct methods** | In-semester Examination: 30 Marks | End Semester Examination: 70 Marks |
| | | | Practical/Oral/Term Work |
| | **Indirect Methods** | Assignments, Presentations | Seminars, Quiz, Q&A session, Group Discussion |
| **Prerequisites** | | | |
| **Course Objectives** | | | |
| 1 | To explore the principles of Object Oriented Programming (OOP). | | |
| 2 | To understand object-oriented concepts such as data abstraction, encapsulation, inheritance, dynamic binding, and polymorphism. | | |
| 3 | To use the object-oriented paradigm in program design. | | |
| 4 | To lay a foundation for advanced programming. | | |
| 5 | Provide programming insight using OOP constructs. | | |
| **Course Outcomes** | | | |
| 1 | Analyze the strengths of object oriented programming. | | |
| 2 | Design and apply OOP principles for effective programming. | | |
| 3 | Develop programming application using object oriented programming language C++ | | |
| 4 | Percept the utility and applicability of OOP | | |
| **Course Contents** | | | |
| **Unit-I** | **Classes and Objects** | | |
| | Need of Object-Oriented Programming (OOP), Object Oriented Programming Paradigm, Basic Concepts of Object-Oriented Programming, Benefits of OOP, C++ as object oriented programming language.<br>**C++ Programming**- C++ programming Basics, Data Types, Structures, Enumerations, control structures, Arrays and Strings, Class, Object, class and data abstraction, class scope and accessing class members, separating interface from implementation, controlling access to members.<br>**Functions-** Function, function prototype, accessing function and utility function, Constructors and destructors, Copy Constructor, Objects and Memory requirements, Static Class members, data abstraction and information hiding, inline function. | | |
| | **Practical/Tutorial** | | |

| | |
|---|---|
| | |
| **Unit-II** | **Polymorphism and Inheritance** |
| | **Operator Overloading-** concept of overloading, operator overloading, Overloading Unary Operators, Overloading Binary Operators, Data Conversion, Type casting (implicit and explicit), Pitfalls of Operator Overloading and Conversion, Keywords explicit and mutable.<br>**Inheritance-** Base Class and derived Class, protected members, relationship between base Class and derived Class, Constructor and destructor in Derived Class, Overriding Member Functions, Class Hierarchies, Inheritance, Public and Private Inheritance, Levels of Inheritance, Multiple Inheritance, Ambiguity in Multiple Inheritance, Aggregation, Classes Within Classes.<br>**Polymorphism-** concept, relationship among objects in inheritance hierarchy, abstract classes, polymorphism. |
| | **Practical/Tutorial** |
| | |
| **Unit-III** | **Virtual Functions** |
| | **Virtual Functions-** Pointers- indirection Operators, Memory Management: new and delete, Pointers to Objects, A Linked List Example, accessing Arrays using pointers, Function pointers, Pointers to Pointers, A Parsing Example, Debugging Pointers, Dynamic Pointers, smart pointers, shared pointers, Case Study : Design of Horse Race Simulation.<br>**Virtual Function-** Friend Functions, Static Functions, Assignment and Copy Initialization, this Pointer, virtual function, dynamic binding, Virtual destructor. |
| | **Practical/Tutorial** |
| **Unit-IV** | **Templates and Exception Handling** |
| | **Templates-** function templates, Function overloading, overloading Function templates, class templates, class template and Nontype parameters, template and inheritance, template and friends<br>Generic Functions, Applying Generic Function, Generic Classes, The typename and export keywords, The Power of Templates.<br>**Exception Handling-** Fundamentals, other error handling techniques, simple exception handling- Divide by Zero, rethrowing an exception, exception specifications, processing unexpected exceptions, stack unwinding, constructor, destructor and exception handling, exception and inheritance. |
| | **Practical/Tutorial** |
| | |
| **Unit- V** | **Files and Streams** |
| | Data hierarchy, Stream and files, Stream Classes, Stream Errors, Disk File I/O with Streams, File Pointers, and Error Handling in File I/O, File I/O with Member Functions, Overloading the Extraction and Insertion Operators, memory as a Stream Object, Command-Line Arguments, Printer output, Early vs. Late Binding. |

| | | | |
|---|---|---|---|
| | **Practical/Tutorial** | | |
| | | | |
| **Unit-VI** | **Standard Template Library (STL)** | | |
| | Introduction to STL, Containers, algorithms and iterators, Containers- Sequence container and associative containers, container adapters, Algorithms- basic searching and sorting algorithms, min-max algorithm, set operations, heap sort, Iterators- input, output, forward, bidirectional and random access. Object Oriented Programming – a road map to future | | |
| | **Practical/Tutorial** | | |
| | | | |
| **Text Books** | **Author** | **Title of Book** | **Publication** |
| T1 | Bjarne Stroustrup | The C++ Programming language | Third edition, Pearson Education. ISBN 9780201889543. |
| T2 | **Deitel** | C++ How to Program | 4th Edition, Pearson Education, ISBN:81-297-0276-2 |
| **Reference Books** | | | |
| R1 | Robert Lafore | Object-Oriented Programming in C++ | fourth edition, Sams Publishing, ISBN:0672323087 (ISBN 13: 9780672323089) |
| R2 | Herbert Schildt | C++ The complete reference | Eighth Edition, McGraw Hill Professional, 2011, ISBN:978-00-72226805 |
| R3 | Matt Weisfeld | The Object-Oriented Thought Process | Third Edition Pearson ISBN-13:075-2063330166 |
| R4 | Cox Brad,<br><br>Andrew J. Novobilski | Object –Oriented Programming: An Evolutionary Approach | Second Edition, Addison–Wesley, ISBN:13:978-020-1548341 |

| | |
|---|---|
| **Self-Learning Facilities, Web Resources, Research papers for reference** | |
| **Contents beyond Syllabus** | |
| **Additional Experiments** | - |
| **Bridging Courses** | - |
| **Tutorials** | - |
| **Presentations** | - |