**II Year - I Semester**

| L | T | P | C |
|---|---|---|---|
| 4 | 0 | 0 | 3 |

## STATISTICS WITH R PROGRAMMING

**OBJECTIVE:**

After taking the course, students will be able to

- Use R for statistical programming, computation, graphics, and modeling,
- Write functions and use R in an efficient way,
- Fit some basic types of statistical models
- Use R in their own research,
- Be able to expand their knowledge of R on their own.

**UNIT-I:**

Introduction, How to run R, R Sessions and Functions, Basic Math, Variables, Data Types, Vectors, Conclusion, Advanced Data Structures, Data Frames, Lists, Matrices, Arrays, Classes**.**

**UNIT-II:**

R Programming Structures, Control Statements, Loops, - Looping Over Nonvector Sets,- If-Else, Arithmetic and Boolean Operators and values, Default Values for Argument, Return Values, Deciding Whether to explicitly call return- Returning Complex Objects, Functions are Objective, No Pointers in R, Recursion, A Quicksort Implementation-Extended Extended Example: A Binary Search Tree.

**UNIT-III:**

Doing Math and Simulation in R, Math Function, Extended Example Calculating Probability- Cumulative Sums and Products-Minima and Maxima- Calculus, Functions Fir Statistical Distribution, Sorting, Linear Algebra Operation on Vectors and Matrices, Extended Example: Vector cross Product- Extended Example: Finding Stationary Distribution of Markov Chains, Set Operation, Input /out put, Accessing the Keyboard and Monitor, Reading and writer Files,

**UNIT-IV:**

Graphics, Creating Graphs, The Workhorse of R Base Graphics, the plot() Function – Customizing Graphs, Saving Graphs to Files.

**UNIT-V:**

Probability Distributions, Normal Distribution- Binomial Distribution- Poisson Distributions Other Distribution, Basic Statistics, Correlation and Covariance, T-Tests,-ANOVA.

**UNIT-VI:**

Linear Models, Simple Linear Regression, -Multiple Regression Generalized Linear Models, Logistic Regression, - Poisson Regression- other Generalized Linear Models-Survival Analysis, Nonlinear Models, Splines- Decision- Random Forests,

**OUTCOMES:**

At the end of this course, students will be able to:

- List motivation for learning a programming language
- Access online resources for R and import new function packages into the R workspace
- Import, review, manipulate and summarize data-sets in R
- Explore data-sets to create testable hypotheses and identify appropriate statistical tests
- Perform appropriate statistical tests using R Create and edit visualizations with

**TEXT BOOKS:**

1) The Art of R Programming, Norman Matloff, Cengage Learning
2) R for Everyone, Lander, Pearson

**REFERENCE BOOKS:**

1) R Cookbook, PaulTeetor, Oreilly.
2) R in Action,Rob Kabacoff, Manning

**II Year - I Semester**

| L | T | P | C |
|---|---|---|---|
| 4 | 0 | 0 | 3 |

## MATHEMATICAL FOUNDATION OF COMPUTER SCIENCE

**OBJECTIVES:**

- To introduce the students to the topics and techniques of discrete methods and combinatorial reasoning.
- To introduce a wide variety of applications. The algorithmic approach to the solution of problems is fundamental in discrete mathematics, and this approach reinforces the close ties between this discipline and the area of computer science.

**UNIT -I:**

**Mathematical Logic**: Propositional Calculus: Statements and Notations, Connectives, Well Formed Formulas, Truth Tables, Tautologies, Equivalence of Formulas, Duality Law, Tautological Implications, Normal Forms, Theory of Inference for Statement Calculus, Consistency of Premises, Indirect Method of Proof. Predicate Calculus:Predicative Logic, Statement Functions, Variables and Quantifiers, Free and Bound Variables, Inference Theory for Predicate Calculus.

**UNIT -II:**

**Set Theory:** Introduction, Operations on Binary Sets, Principle of Inclusion and Exclusion, *Relations:* Properties of Binary Relations, Relation Matrix and Digraph, Operations on Relations, Partition and Covering, Transitive Closure, Equivalence, Compatibility and Partial Ordering Relations, Hasse Diagrams, *Functions:* Bijective Functions, Composition of Functions, Inverse Functions, Permutation Functions, Recursive Functions, Lattice and its Properties.

**UNIT- III:**

**Algebraic Structures and Number Theory:** *Algebraic Structures:*Algebraic Systems, Examples, General Properties, Semi Groups and Monoids, Homomorphism of Semi Groups and Monoids, Group, Subgroup, Abelian Group, Homomorphism, Isomorphism, *Number Theory:*Properties of Integers, Division Theorem, The Greatest Common Divisor, Euclidean Algorithm, Least Common Multiple, Testing for Prime Numbers, The Fundamental Theorem of Arithmetic, Modular Arithmetic (Fermat's Theorem and Euler's Theorem)

**UNIT -IV:**

**Combinatorics:** Basic of Counting, Permutations, Permutations with Repetitions, Circular Permutations, Restricted Permutations, Combinations, Restricted Combinations, Generating Functions of Permutations and Combinations, Binomial and Multinomial Coefficients, Binomial and Multinomial Theorems, The Principles of Inclusion–Exclusion, Pigeonhole Principle and its Application.

**UNIT -V:**

**Recurrence Relations:** Generating Functions, Function of Sequences, Partial Fractions, Calculating Coefficient of Generating Functions, Recurrence Relations, Formulation as Recurrence Relations, Solving Recurrence Relations by Substitution and Generating Functions, Method of Characteristic Roots, Solving Inhomogeneous Recurrence Relations

**UNIT -VI:**

**Graph Theory:** Basic Concepts of Graphs, Sub graphs, Matrix Representation of Graphs: Adjacency Matrices, Incidence Matrices, Isomorphic Graphs, Paths and Circuits, Eulerian and Hamiltonian Graphs, Multigraphs, Planar Graphs, Euler's Formula, Graph Colouring and Covering, Chromatic Number, Spanning Trees, Algorithms for Spanning Trees (Problems Only and Theorems without Proofs).

**OUTCOMES:**

- Student will be able to demonstrate skills in solving mathematical problems
- Student will be able to comprehend mathematical principles and logic
- Student will be able to demonstrate knowledge of mathematical modeling and proficiency in using mathematical software
- Student will be able to manipulate and analyze data numerically and/or graphically using appropriate Software
- Student will be able to communicate effectively mathematical ideas/results verbally or in writing

**TEXT BOOKS:**

1.Discrete Mathematical Structures with Applications to Computer Science, J. P. Tremblay and P. Manohar, Tata McGraw Hill.

2. Elements of Discrete Mathematics-A Computer Oriented Approach, C. L. Liu and D. P. Mohapatra, 3rd Edition, Tata McGraw Hill.

3. Discrete Mathematics and its Applications with Combinatorics and Graph Theory, K. H. Rosen, 7th Edition, Tata McGraw Hill.

**REFERENCE BOOKS:**

1. Discrete Mathematics for Computer Scientists and Mathematicians, J. L. Mott, A. Kandel, T.P. Baker, 2nd Edition, Prentice Hall of India.

2. Discrete Mathematical Structures, BernandKolman, Robert C. Busby, Sharon Cutler Ross, PHI.

3. Discrete Mathematics, S. K. Chakraborthy and B.K. Sarkar, Oxford, 2011.

**II Year - I Semester**

| L | T | P | C |
|---|---|---|---|
| 4 | 0 | 0 | 3 |

# DIGITAL LOGIC DESIGN

**OBJECTIVE:**

- To introduce the basic tools for design with combinational and sequential digital logic and state machines.
- To learn simple digital circuits in preparation for computer engineering.

**UNIT- I: Digital Systems and Binary Numbers**

Digital Systems, Binary Numbers, Binary Numbers, Octal and Hexadecimal Numbers, Complements of Numbers, Complements of Numbers, Signed Binary Numbers,Arithmetic addition and subtraction

**UNIT -II: Concept of Boolean algebra**

Basic Theorems and Properties of Boolean algebra, Boolean Functions, Canonical and Standard Forms, Minterms and Maxterms,

**UNIT- III: Gate level Minimization**

Map Method, Two-Variable K-Map, Three-Variable K-Map, Four Variable K-Maps. Products of Sum Simplification, Sum of Products Simplification, Don't – Care Conditions, NAND and NOR Implementation, Exclusive-OR Function

**UNIT- IV:Combinational Logic**

Introduction, Analysis Procedure, Design Procedure, Binary Adder–Subtractor, Decimal Adder, Binary Multiplier, Decoders, Encoders, Multiplexers, HDL Models of Combinational Circuits

**UNIT- V: Synchronous Sequential Logic**

Introduction to Sequential Circuits, Storage Elements: Latches, Storage Elements: Flip-Flops, Analysis of Clocked **Sequential** Circuits, Mealy and Moore Models of Finite State Machines

**UNIT -VI: Registers and Counters**

Registers, Shift Registers, Ripple Counters, Synchronous Counters, Ring Counter, Johnson Counter, Ripple Counter

**OUTCOMES:**

A student who successfully fulfills the course requirements will have demonstrated:

- An ability to define different number systems, binary addition and subtraction, 2's complement representation and operations with this representation.
- An ability to understand the different switching algebra theorems and apply them for logic functions.
- An ability to define the Karnaugh map for a few variables and perform an algorithmic reduction of logic functions.
- An ability to define the other minimization methods for any number of variables Variable Entered Mapping (VEM) and Quine-MeCluskey (QM) Techniques and perform an algorithmic reduction of logic functions.

**TEXT BOOKS:**

1. Digital Design, 5/e, M.Morris Mano, Michael D Ciletti, PEA.
2. Fundamentals of Logic Design, 5/e, Roth, Cengage.

**REFERENCE BOOKS:**

1. Digital Logic and Computer Design, M.Morris Mano, PEA.
2. Digital Logic Design, Leach, Malvino, Saha, TMH.
3. Modern Digital Electronics, R.P. Jain, TMH.

**II Year - I Semester**

| L | T | P | C |
|---|---|---|---|
| 4 | 0 | 0 | 3 |

# PYTHON PROGRAMMING

**OBJECTIVES:**

- Introduction to Scripting Language
- Exposure to various problems solving approaches of computer science

**UNIT – I:**

**Introduction:**History of Python, Need of Python Programming, Applications Basics of Python Programming Using the REPL(Shell), Running Python Scripts, Variables, Assignment, Keywords, Input-Output, Indentation.

**UNIT – II:**

**Types, Operators and Expressions:** Types - Integers, Strings, Booleans; Operators- Arithmetic Operators, Comparison (Relational) Operators, Assignment Operators, Logical Operators, Bitwise Operators, Membership Operators, Identity Operators, Expressions and order of evaluations Control Flow- if, if-elif-else, for, while, break, continue, pass

**UNIT – III:**

**Data Structures** Lists - Operations, Slicing, Methods; Tuples, Sets, Dictionaries, Sequences. Comprehensions.

**UNIT – IV:**

**Functions** - Defining Functions, Calling Functions, Passing Arguments, Keyword Arguments, Default Arguments, Variable-length arguments, Anonymous Functions, Fruitful Functions(Function Returning Values), Scope of the Variables in a Function - Global and Local Variables.

**Modules**: Creating modules, import statement, from. Import statement, name spacing,

**Python packages**, Introduction to PIP, Installing Packages via PIP, Using Python Packages

**UNIT – V:**

**Object Oriented Programming OOP in Python**: Classes, 'self variable', Methods, Constructor Method, Inheritance, Overriding Methods, Datahiding,

**Error and Exceptions:** Difference between an error and Exception, Handling Exception, try except block, Raising Exceptions, User Defined Exceptions

**UNIT – VI:**

**Brief Tour of the Standard Library** - Operating System Interface - String Pattern Matching, Mathematics, Internet Access, Dates and Times, Data Compression, Multithreading, GUI Programming, Turtle Graphics

**Testing:** Why testing is required ?, Basic concepts of testing, Unit testing in Python, Writing Test cases, Running Tests.

**OUTCOMES:**

- Making Software easily right out of the box.
- Experience with an interpreted Language.
- To build software for real needs.
- Prior Introduction to testing software

**TEXT BOOKS**

1. Python Programming: A Modern Approach, Vamsi Kurama, Pearson
2. Learning Python, Mark Lutz, Orielly

**Reference Books:**

1. Think Python, Allen Downey, Green Tea Press
2. Core Python Programming, W.Chun, Pearson.
3. Introduction to Python, Kenneth A. Lambert, Cengage

**II Year - I Semester**

| L | T | P | C |
|---|---|---|---|
| 4 | 0 | 0 | 3 |

## DATA STRUCTURES THROUGH C++

**OBJECTIVES:**

- To be familiar with basic techniques of object oriented principles and exception handling using C++
- To be familiar with the concepts like Inheritance, Polymorphism
- Solve problems using data structures such as linear lists, stacks, queues, hash tables
- Be familiar with advanced data structures such as balanced search trees, AVLTrees, and B Trees.

**UNIT-I: ARRAYS**

Abstract Data Types and the C++ Class, An Introduction to C++ Class- Data Abstraction and Encapsulation in C++- Declaring Class Objects and Invoking Member Functions- Special Class Operations- Miscellaneous Topics- ADTs and C++Classes, The Array as an Abstract Data Type, The Polynomial Abstract Data type- Polynomial Representation- Polynomial Addition. Spares Matrices,Introduction- Sparse Matrix Representation- Transposing a Matrix- Matrix Multiplication, Representation of Arrays.

**UNIT-II: STACKS AND QUEUES**

Templates in C++, Template Functions- Using Templates to Represent Container Classes, The Stack Abstract Data Type, The Queue Abstract Data Type, Subtyping and Inheritance in C++, Evaluation of Expressions, Expression- Postfix Notation- Infix to Postfix.

**UNIT-III: LINKED LISTS**

Single Linked List and Chains, Representing Chains in C++, Defining a Node in C++- Designing a Chain Class in C++- Pointer manipulation in C++- Chain Manipulation Operations, The Template Class Chain, Implementing Chains with Templates- Chain Iterators- Chain Operations- Reusing a Class, Circular Lists, Available Space Lists, Linked Stacks and Queues, Polynomials, Polynomial Representation- Adding Polynomials- Circular List Representation of Polynomials, Equivalence Classes, Sparse Matrices, Sparse Matrix Representation- Sparse Matrix Input- Deleting a Sparse Matrix, Doubly Linked Lists, Generalized Lists, Representation of Generalized Lists- Recursive Algorithms for Lists- Reference Counts, Shared and Recursive Lists

**UNIT-IV: TREES**

Introduction, Terminology, Representation of Trees, Binary Trees, The Abstract Data Type, Properties of Binary Tress, Binary Tree Representations, Binary Tree Traversal and Tree Iterators, Introduction, Inorder Traversal Preorder Traversal, Postorder Traversal, Thread Binary Trees, Threads, Inorder Traversal of a Threaded Binary Tree, Inserting a Node into a Threaded Binary Tree, Heaps, Priority Queues, Definition of a Max Heap, Insertion into a Max Heap, Deletion from a Max Heap, Binary Search Trees, Definition, Searching a Binary Search Tree, Insertion into a Binary Search Tree, Deletion from a Binary Search Tree, Height of Binary Search Tree.

**UNIT-V: GRAPHS**

The Graph Abstract Data Type, Introduction, Definition, Graph Representation, Elementary Graph Operation, Depth First Search, Breadth First Search, Connected Components, Spanning Trees, Biconnected Components, Minimum Cost Spanning Trees, Kruskal S Algorithm, Prim s Algorithm Sollin' s Algorithm, Shortest Paths and Transitive Closure, Single Source/All Destination: Nonnegative Edge Cost, Single Source/All Destination: General Weights, All-Pairs Shortest Path, Transitive Closure.

**UNIT-VI: SORTING**

Insertion Sort, Quick Sort, Merge Sort Merging, Iterative Merge Sort, Recursive Merge Sort, Heap Sort.

**OUTCOMES:**

- Distinguish between procedures and object oriented programming.
- Apply advanced data structure strategies for exploring complex data structures.
- Compare and contrast various data structures and design techniques in the area of Performance.
- Implement data structure algorithms through C++. • Incorporate data structures into the applications such as binary search trees, AVL and B Trees
- Implement all data structures like stacks, queues, trees, lists and graphs and compare their Performance and trade offs

**TEXT BOOKS:**

1. Data structures, Algorithms and Applications in C++, S.Sahni, University Press (India) Pvt.Ltd, 2nd edition, Universities Press, Pvt. Ltd.
2. Data structures and Algorithm Analysis in C++, Mark Allen Weiss, Pearson Education. Ltd., Second Edition.
3. Data structures and Algorithms in C++, Michael T.Goodrich, R.Tamassia and .Mount, Wiley student edition, John Wiley and Sons.

**REFERENCE BOOKS:**

1. Data structures and algorithms in C++, 3rd Edition, Adam Drozdek, Thomson
2. Data structures using C and C++, Langsam, Augenstein and Tanenbaum, PHI.
3. Problem solving with C++, The OOP, Fourth edition, W.Savitch, Pearson education.

**II Year - I Semester**

| L | T | P | C |
|---|---|---|---|
| 4 | 0 | 0 | 3 |

## COMPUTER GRAPHICS

**OBJECTIVES:**

- To develop, design and implement two and three dimensional graphical structures
- To enable students to acquire knowledge Multimedia compression and animations
- To learn Creation, Management and Transmission of Multimedia objects.

**UNIT-I:**
**2D Primitives** Output primitives – Line, Circle and Ellipse drawing algorithms - Attributes of output primitives – Two dimensional Geometric transformations - Two dimensional viewing – Line, Polygon, Curve and Text clipping algorithms

**UNIT-II:**
**3D Concepts** Parallel and Perspective projections - Three dimensional object representation – Polygons, Curved lines, Splines, Quadric Surfaces, - Visualization of data sets - 3Dtransformations – Viewing -Visible surface identification.

**UNIT-III:**
**Graphics Programming**Color Models – RGB, YIQ, CMY, HSV – Animations – General Computer Animation, Raster, Keyframe - Graphics programming using OPENGL – Basic graphics primitives –Drawing three dimensional objects - Drawing three dimensional scenes

**UNIT- IV:**
**Rendering** Introduction to Shading models – Flat and Smooth shading – Adding texture to faces –Adding shadows of objects – Building a camera in a program – Creating shaded objects– Rendering texture – Drawing Shadows.

**UNIT- V:**
**Fractals**Fractals and Self similarity – Peano curves – Creating image by iterated functions – Mandelbrot sets – Julia Sets – Random Fractals

**UNIT- VI:**

**Overview of Ray Tracing** Intersecting rays with other primitives – Adding Surface texture – Reflections andTransparency – Boolean operations on Objects.

**OUTCOMES:**

- Know and be able to describe the general software architecture of programs that use 3D computer graphics.
- Know and be able to discuss hardware system architecture for computer graphics. This Includes, but is not limited to: graphics pipeline, frame buffers, and graphic accelerators/co-processors.
- Know and be able to select among models for lighting/shading: Color, ambient light; distant and light with sources; Phong reflection model; and shading (flat, smooth, Gourand, Phong).

**TEXT BOOKS:**

1. Donald Hearn, Pauline Baker, Computer Graphics – C Version, second edition Pearson Education,2004.
2. F.S. Hill, Computer Graphics using OPENGL, Second edition, Pearson Education, 2003.

**REFERENCE BOOKS:**

1. James D. Foley, Andries Van Dam, Steven K. Feiner, John F. Hughes, Computer Graphics- Principles and practice, Second Edition in C, Pearson Education, 2007.

**II Year - I Semester**

| L | T | P | C |
|---|---|---|---|
| 0 | 0 | 3 | 2 |

## DATASTRUCTURES THROUGH C++ LAB

**OBJECTIVES:**

- To develop skills to design and analyze simple linear and non linear data structures
- To Strengthen the ability to identify and apply the suitable data structure for the given real world problem
- To Gain knowledge in practical applications of data structures

**List of Experiments:**

1. Implementation of Singly linked list.
2. Implementation of Doubly linked list.
3. Implementation of Multistack in a Single Array.
4. Implementation of Circular Queue
5. Implementation of Binary Search trees.
6. Implementation of Hash table.
7. Implementation of Heaps.
8. Implementation of Breadth First Search Techniques.
9. Implementation of Depth First Search Techniques.
10. Implementation of Prim's Algorithm.
11. Implementation of Dijkstra's Algorithm.
12. Implementation of Kruskal's Algorithm
13. Implementation of MergeSort
14. Implementation of Quick Sort
15. Implementation of Data Searching using divide and conquer technique

**OUTCOMES:**

At the end of this lab session, the student will

- Be able to design and analyze the time and space efficiency of the data structure
- Be capable to identity the appropriate data structure for given problem

Have practical knowledge on the application of data structures

**II Year - I Semester**

| L | T | P | C |
|---|---|---|---|
| 0 | 0 | 3 | 2 |

## PYTHON PROGRAMMING LAB

### Exercise 1 - Basics

a) Running instructions in Interactive interpreter and a Python Script
b) Write a program to purposefully raise Indentation Error and Correct it

### Exercise 2 - Operations

a) Write a program to compute distance between two points taking input from the user (Pythagorean Theorem)
b) Write a program add.py that takes 2 numbers as command line arguments and prints its sum.

### Exercise - 3 Control Flow

a) Write a Program for checking whether the given number is a even number or not.

b) Using a for loop, write a program that prints out the decimal equivalents of 1/2, 1/3, 1/4, . . . , 1/10

c) Write a program using a for loop that loops over a sequence. What is sequence ?

d) Write a program using a while loop that asks the user for a number, and prints a countdown from that number to zero.

### Exercise 4 - Control Flow - Continued

a) Find the sum of all the primes below two million.
Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:

1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

b) By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.

### Exercise - 5 - DS

a) Write a program to count the numbers of characters in the string and store them in a dictionary data structure
b) Write a program to use split and join methods in the string and trace a birthday with a dictionary data structure.

**Exercise - 6 DS - Continued**

a) Write a program combine_lists that combines these lists into a dictionary.

b) Write a program to count frequency of characters in a given file. Can you use character frequency to tell whether the given file is a Python program file, C program file or a text file?

**Exercise - 7 Files**

a) Write a program to print each line of a file in reverse order.
b) Write a program to compute the number of characters, words and lines in a file.

**Exercise - 8 Functions**

a) Write a function ball_collide that takes two balls as parameters and computes if they are colliding. Your function should return a Boolean representing whether or not the balls are colliding.

   Hint: Represent a ball on a plane as a tuple of (x, y, r), r being the radius

   If (distance between two balls centers) <= (sum of their radii) then (they are colliding)

b) Find mean, median, mode for the given set of numbers in a list.

**Exercise - 9 Functions - Continued**

a) Write a function nearly_equal to test whether two strings are nearly equal. Two strings a and b are nearly equal when a can be generated by a single mutation on b.
b) Write a function dups to find all duplicates in the list.
c) Write a function unique to find all the unique elements of a list.

**Exercise - 10 - Functions - Problem Solving**

a) Write a function cumulative_product to compute cumulative product of a list of numbers.
b) Write a function reverse to reverse a list. Without using the reverse function.
c) Write function to compute gcd, lcm of two numbers. Each function shouldn't exceed one line.

**Exercise 11 - Multi-D Lists**

a) Write a program that defines a matrix and prints
b) Write a program to perform addition of two square matrices
c) Write a program to perform multiplication of two square matrices

**Exercise - 12 - Modules**

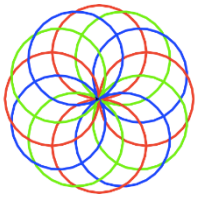
a) Install packages requests, flask and explore them. using (pip)
b) Write a script that imports requests and fetch content from the page. Eg. (Wiki)
c) Write a simple script that serves a simple HTTPResponse and a simple HTML Page

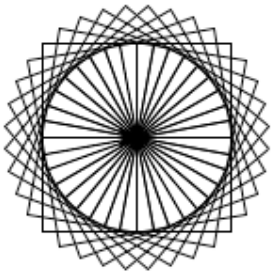
**Exercise - 13 OOP**

a) Class variables and instance variable and illustration of the self variable
   i) Robot
   ii) ATM Machine

**Exercise - 14 GUI, Graphics**

1. Write a GUI for an Expression Calculator using tk
2. Write a program to implement the following figures using turtle





**Exercise - 15 - Testing**

a) Write a test-case to check the function even_numbers which return True on passing a list of all even numbers
b) Write a test-case to check the function reverse_string which returns the reversed string

**Exercise - 16 - Advanced**

a) Build any one classical data structure.
b) Write a program to solve knapsack problem.

**II Year – II Semester**

| L | T | P | C |
|---|---|---|---|
| 4 | 0 | 0 | 3 |

## SOFTWARE ENGINEERING

**OBJECTIVES**

- To understand the software life cycle models.
- To understand the software requirements and SRS document.
- To understand the importance of modeling and modeling languages.
- To design and develop correct and robust software products.
- To understand the quality control and how to ensure good quality software.
- To understand the planning and estimation of software projects.
- To understand the implementation issues, validation and verification procedures.
- To understand the maintenance of software

**UNIT-I:**

**Software and Software Engineering:** The Nature of Software, The Unique Nature of WebApps, Software Engineering, Software Process, Software Engineering Practice, Software Myths.

**Process Models:** A Generic Process Model, Process Assessment and Improvement, Prescriptive Process Models, Specialized Process Models, The Unified Process, Personal and Team Process Models, Process Terminology, Product and Process.

**UNIT-II:**

**Requirements Analysis And Specification:** Requirements Gathering and Analysis, Software Requirement Specification (SRS), Formal System Specification.

**Software Design:** Overview of the Design Process, How to Characterise of a Design?, Cohesion and Coupling, Layered Arrangement of Modules, Approaches to Software Design

**UNIT – III:**

**Function-Oriented Software Design:** Overview of SA/SD Methodology, Structured Analysis, Developing the DFD Model of a System, Structured Design, Detailed Design, Design Review, over view of Object Oriented design.

**User Interface Design**: Characteristics of Good User Interface, Basic Concepts, Types of User Interfaces, Fundamentals of Component-based GUI Development, A User Interface Design Methodology.

**UNIT – IV:**

**Coding And Testing:** Coding, Code Review, Software Documentation, Testing, Unit Testing, Black-Box Testing, White-Box Testing, Debugging, Program Analysis Tool, Integration Testing, Testing Object-Oriented Programs, System Testing, Some General Issues Associated with Testing

**UNIT – V:**

**Software Reliability And Quality Management:** Software Reliability, Statistical Testing, Software Quality, Software Quality Management System, ISO 9000, SEI Capability Maturity Model.

**Computer Aided Software Engineering:** Case and its Scope, Case Environment, Case Support in Software Life Cycle, Other Characteristics of Case Tools, Towards Second Generation CASE Tool, Architecture of a Case Environment

**UNIT – VI**

**Software Maintenance:** Software maintenance, Maintenance Process Models, Maintenance Cost, Software Configuration Management.

**Software Reuse**: what can be reused? Why almost No Reuse So Far? Basic Issues in Reuse Approach, Reuse at Organization Level.

**OUTCOMES**

- Define and develop a software project from requirement gathering to implementation.
- Obtain knowledge about principles and practices of software engineering.
- Focus on the fundamentals of modeling a software project.
- Obtain knowledge about estimation and maintenance of software systems

**TEXT BOOKS:**


1. Software engineering A practitioner's Approach, Roger S. Pressman, Seventh Edition McGrawHill International Edition.
2. Fundamentals of Software Engineering, Rajib Mall, Third Edition, PHI.
3. Software Engineering, Ian Sommerville, Ninth edition, Pearson education


**REFERENCE BOOKS:**


1. Software Engineering : A Primer, Waman S Jawadekar, Tata McGraw-Hill, 2008
2. Software Engineering, A Precise Approach, Pankaj Jalote, Wiley India, 2010.
3. Software Engineering, Principles and Practices, Deepak Jain, Oxford University Press.
4. Software Engineering1: Abstraction and modeling, Diner Bjorner, Springer International edition, 2006.

**II Year - II Semester**

| L | T | P | C |
|---|---|---|---|
| 4 | 0 | 0 | 3 |

# JAVA PROGRAMMING

**OBJECTIVES:**

- Understanding the OOP's concepts, classes and objects, threads, files, applets, swings and act.
- This course introduces computer programming using the JAVA programming language with object-oriented programming principles.
- Emphasis is placed on event-driven programming methods, including creating and manipulating objects, classes, and using Java for network level programming and middleware development

**UNIT-I:**

Introduction to OOP, procedural programming language and object oriented language, principles of OOP, applications of OOP, history of java, java features, JVM, program structure.

Variables, primitive data types, identifiers, literals, operators, expressions, precedence rules and associativity, primitive type conversion and casting, flow of control.

**UNIT-II:**

Classes and objects, class declaration, creating objects, methods, constructors and constructor overloading, garbage collector, importance of static keyword and examples, this keyword, arrays, command line arguments, nested classes.

**UNIT-III:**

Inheritance, types of inheritance, super keyword, final keyword, overriding and abstract class.

Interfaces, creating the packages, using packages, importance of CLASSPATH and java.lang package. Exception handling, importance of try, catch, throw, throws and finally block, user-defined exceptions, Assertions.

**UNIT-IV:**

Multithreading: introduction, thread life cycle, creation of threads, thread priorities, thread synchronization, communication between threads. Reading data from files and writing data to files, random access file,

**UNIT-V:**

Applet class, Applet structure, Applet life cycle, sample Applet programs. Event handling: event delegation model, sources of event, Event Listeners, adapter classes, inner classes.

**UNIT-VI:**

AWT: introduction, components and containers, Button, Label, Checkbox, Radio Buttons, List Boxes, Choice Boxes, Container class, Layouts, Menu and Scrollbar.

**OUTCOMES:**

- Understand Java programming concepts and utilize Java Graphical User Interface in Program writing.
- Write, compile, execute and troubleshoot Java programming for networking concepts.
- Build Java Application for distributed environment.
- Design and Develop multi-tier applications.
- Identify and Analyze Enterprise applications.

**TEXT BOOKS:**


1. The complete Reference Java, 8th edition, Herbert Schildt, TMH.
2. Programming in JAVA, Sachin Malhotra, SaurabhChoudary, Oxford.
3. Introduction to java programming, 7th edition by Y Daniel Liang, Pearson.


**REFERENCE BOOKS:**

1. Swing: Introduction, JFrame, JApplet, JPanel, Componets in Swings, Layout Managers in
2. Swings, JList and JScrollPane, Split Pane, JTabbedPane, JTree, JTable, Dialog Box.

**II Year - II Semester**

| L | T | P | C |
|---|---|---|---|
| 4 | 0 | 0 | 3 |

## ADVANCED DATA STRUCTURES

**OBJECTIVES:**

- Describe and implement a variety of advanced data structures (hash tables, priority queues, balanced search trees, graphs).
- Analyze the space and time complexity of the algorithms studied in the course.
- Identify different solutions for a given problem; analyze advantages and disadvantages to different solutions.
- Demonstrate an understanding of external memory and external search and sorting algorithms.
- Demonstrate an understanding of simple Entity-Relationship models for databases.

**UNIT-I: SORTING**

External Sorting, Introduction, K-way Merging - Buffer Handling for parallel Operation- Run Generation- Optimal Merging of Runs.

**UNIT-II: HASHING**

Introduction-Static Hashing- Hash Table- Hash Functions- Secure Hash Function- Overflow Handling- Theoretical Evaluation of Overflow Techniques, Dynamic Hashing- Motivation for Dynamic Hashing -Dynamic Hashing Using Directories- Directory less Dynamic, Hashing,

**UNIT-III:PRIORITY QUEUES (HEAPS)**

Model, Simple Implementation, Binary Heap-Structure Property-Heap-Order Property-Basic Heap Operations- Other Heap Operation, Applications of Priority Queues- The Selection Problem Event Simulation Problem, Binomial Queues- Binomial Queue Structure – Binomial Queue Operation- Implementation of Binomial Queues

**UNIT-IV: EFFICIENT BINARY SEARCH TREES**

Optimal Binary Search Trees, AVL Trees, Red-Black Trees, Definition- Representation of a Red- Black Tree- Searching a Red-Black Tree- Inserting into a Red Black Tree- Deletion from a Red-Black Tree- Joining Red-Black Trees, Splitting a Red-Black tree.

**UNIT-V: MULTIWAY SEARCH TREES**

M-Way Search Trees, Definition and Properties- Searching an M-Way Search Tree, B-Trees, Definition and Properties- Number of Elements in a B-tree- Insertion into B-Tree- Deletion from a B-Tree- B+-Tree Definition- Searching a B+-Tree- Insertion into B+-tree- Deletion from a B+-Tree.

**UNIT-VI: DIGITAL SEARCH STRUCTURES**

Digital Search Trees, Definition- Search, Insert and Delete- Binary tries and Patricia, Binary Tries, Compressed Binary Tries- Patricia, Multiway Tries- Definitions- Searching a Trie- Sampling Strategies- Insertion into a Trie- Deletion from a Trie- Keys with Different Length- Height of a Trie- Space Required and Alternative Node Structure- Prefix Search and Applications- Compressed Tries- Compressed Tries With Skip Fields- Compressed Tries With Labeled Edges- Space Required by a Compressed Tries, Tries and Internet Packet Forwarding ,- IP Routing- 1-Bit Tries- Fixed-Stride Tries-Variable-Stride Tries.

**OUTCOMES:**

- Be able to understand and apply amortised analysis on data structures, including binary search trees, mergable heaps, and disjoint sets.
- Understand the implementation and complexity analysis of fundamental algorithms such as RSA, primality testing, max flow, discrete Fourier transform.
- Have an idea of applications of algorithms in a variety of areas, including linear programming and duality, string matching, game-theory

**TEXT BOOKS:**


1. Data Structures, a Pseudocode Approach, Richard F Gilberg, Behrouz A Forouzan, Cengage.
2. Fundamentals of DATA STRUCTURES in C: 2nd ed, , Horowitz , Sahani, Anderson-freed, Universities Press
3. Data structures and Algorithm Analysis in C, 2nd edition, Mark Allen Weiss, Pearson


**REFERENCE BOOKS:**


1. Web : http://lcm.csa.iisc.ernet.in/dsa/dsa.html
2. http://utubersity.com/?page_id=878
3. http://freevideolectures.com/Course/2519/C-Programming-and-Data-Structures
4. http://freevideolectures.com/Course/2279/Data-Structures-And-Algorithms
5. File Structures :An Object oriented approach with C++, 3rd ed, Michel J Folk, Greg Riccardi, Bill Zoellick
6. C and Data Structures: A Snap Shot oriented Treatise with Live examples from Science and Engineering, NB Venkateswarlu & EV Prasad, S Chand, 2010.

**II Year - II Semester**

| L | T | P | C |
|---|---|---|---|
| 4 | 0 | 0 | 3 |

## COMPUTER ORGANIZATION

**OBJECTIVES:**

- Understand the architecture of a modern computer with its various processing units. Also the Performance measurement of the computer system.
- In addition to this the memory management system of computer.

**UNIT -I:**

**Basic Structure Of Computers:** Functional unit, Basic Operational concepts, Bus structures, System Software, Performance, The history of computer development.

**UNIT -II:**

**Machine Instruction and Programs:**

Instruction and Instruction Sequencing: Register Transfer Notation, Assembly Language Notation, Basic Instruction Types,

Addressing Modes, Basic Input/output Operations, The role of Stacks and Queues in computer programming equation. Component of Instructions: Logic Instructions, shift and Rotate Instructions

**UNIT -III:**

**Type of Instructions:** Arithmetic and Logic Instructions, Branch Instructions, Addressing Modes, Input/output Operations

**UNIT -IV:**

**INPUT/OUTPUT ORGANIZATION:** Accessing I/O Devices, Interrupts: Interrupt Hardware, Enabling and Disabling Interrupts, Handling Multiple Devices, Direct Memory Access,

Buses: Synchronous Bus, Asynchronous Bus, Interface Circuits, Standard I/O Interface: Peripheral Component Interconnect (PCI) Bus, Universal Serial Bus (USB)

**UNIT -V:**

**The MEMORY SYSTEMS:** Basic memory circuits, Memory System Consideration, Read-Only Memory: ROM, PROM, EPROM, EEPROM, Flash Memory,

Cache Memories: Mapping Functions, INTERLEAVING

**Secondary Storage:** Magnetic Hard Disks, Optical Disks,

**UNIT -VI:**

**Processing Unit:** Fundamental Concepts: Register Transfers, Performing An Arithmetic Or Logic Operation, Fetching A Word From Memory,

Execution of Complete Instruction, Hardwired Control,

**Micro programmed Control:** Microinstructions, Micro program Sequencing, Wide Branch Addressing Microinstructions with next –Address Field

**OUTCOMES:**

- Students can understand the architecture of modern computer.
- They can analyze the Performance of a computer using performance equation
- Understanding of different instruction types.
- Students can calculate the effective address of an operand by addressing modes
- They can understand how computer stores positive and negative numbers.
- Understanding of how a computer performs arithmetic operation of positive and negative numbers.

**TEXT BOOKS:**

1. Computer Organization, Carl Hamacher, Zvonks Vranesic, Safea Zaky, 5th Edition, McGraw Hill.
2. Computer Architecture and Organization, John P. Hayes, 3rd Edition, McGraw Hill.

**REFERENCE BOOKS:**

1. Computer Organization and Architecture – William Stallings Sixth Edition, Pearson/PHI
2. Structured Computer Organization – Andrew S. Tanenbaum, 4th Edition PHI/Pearson
3. Fundamentals or Computer Organization and Design, - Sivaraama Dandamudi Springer Int. Edition.
4. “Computer Organization and Design: The Hardware/Software Interface” by David A. Patterson and John L. Hennessy.
5. J .P. Hayes, "Computer Architecture and Organization", McGraw-Hill, 1998.

**II Year – II Semester**

| L | T | P | C |
|---|---|---|---|
| 4 | 0 | 0 | 3 |

# FORMAL LANGUAGE AND AUTOMATA THEORY

**OBJECTIVE:**

- Introduce the student to the concepts of Theory of computation in computer science
- The students should acquire insights into the relationship among formal languages, formal Grammars and automat.

**UNIT – I: Finite Automata**

Why Study Automata Theory? The Central Concepts of Automata Theory, Automation, Finite Automation, Transition Systems, Acceptance of a String by a Finite Automation, DFA, Design of DFAs, NFA, Design of NFA, Equivalence of DFA and NFA, Conversion of NFA into DFA, Finite Automata with E-Transition, Minimization of Finite Automata, Mealy and Moore Machines, Applications and Limitation of Finite Automata.

**UNIT – II: Regular Expressions**

Regular Expressions, Regular Sets, Identity Rules, Equivalence of two Regular Expressions, Manipulations of Regular Expressions, Finite Automata, and Regular Expressions, Inter Conversion, Equivalence between Finite Automata and Regular Expressions, Pumping Lemma, Closers Properties, Applications of Regular Expressions, Finite Automata and Regular Grammars, Regular Expressions and Regular Grammars.

**UNIT – III: Context Free Grammars**

Formal Languages, Grammars, Classification of Grammars, Chomsky Hierarchy Theorem, Context Free Grammar, Leftmost and Rightmost Derivations, Parse Trees, Ambiguous Grammars, Simplification of Context Free Grammars-Elimination of Useless Symbols, E-Productions and Unit Productions, Normal Forms for Context Free Grammars-Chomsky Normal Form and Greibach Normal Form, Pumping Lemma, Closure Properties, Applications of Context Free Grammars.

**UNIT – IV: Pushdown Automata**

Pushdown Automata, Definition, Model, Graphical Notation, Instantaneous Description Language Acceptance of pushdown Automata, Design of Pushdown Automata, Deterministic and Non – Deterministic Pushdown Automata, Equivalence of Pushdown Automata and Context Free Grammars Conversion, Two Stack Pushdown Automata, Application of Pushdown Automata.

**UNIT – V: Turning Machine**

Turing Machine, Definition, Model, Representation of Turing Machines-Instantaneous Descriptions, Transition Tables and Transition Diagrams, Language of a Turing Machine, Design of Turing Machines, Techniques for Turing Machine Construction, Types of Turing Machines, Church's Thesis, Universal Turing Machine, Restricted Turing Machine.

**UNIT – VI: Computability**

Decidable and Un-decidable Problems, Halting Problem of Turing Machines, Post's Correspondence Problem, Modified Post's Correspondence Problem, Classes of P and NP, NP-Hard and NP-Complete Problems.

**OUTCOMES:**

- Classify machines by their power to recognize languages,
- Employ finite state machines to solve problems in computing,
- Explain deterministic and non-deterministic machines,
- Comprehend the hierarchy of problems arising in the computer science

**TEXT BOOKS:**

1. Introduction to Automata Theory, Languages and Computation, J.E.Hopcroft, R.Motwani and J.D.Ullman, 3rd Edition, Pearson, 2008.
2. Theory of Computer Science-Automata, Languages and Computation, K.L.P.Mishra and N.Chandrasekharan, 3rd Edition, PHI, 2007.

**REFERENCE BOOKS:**

1. Formal Language and Automata Theory, K.V.N.Sunitha and N.Kalyani, Pearson, 2015.
2. Introduction to Automata Theory, Formal Languages and Computation, Shyamalendu Kandar, Pearson, 2013.
3. Theory of Computation, V.Kulkarni, Oxford University Press, 2013.
4. Theory of Automata, Languages and Computation, Rajendra Kumar, McGraw Hill, 2014.

**II Year – II Semester**

| L | T | P | C |
|---|---|---|---|
| 4 | 0 | 0 | 3 |

## PRINCIPLES OF PROGRAMMING LANGUAGES

**OBJECTIVES:**

- To understand and describe syntax and semantics of programming languages
- To understand data, data types, and basic statements
- To understand call-return architecture and ways of implementing them
- To understand object-orientation, concurrency, and event handling in programming languages
- To develop programs in non-procedural programming paradigms

**UNIT- I:**

**Syntax and semantics:** Evolution of programming languages, describing syntax, context, free grammars, attribute grammars, describing semantics, lexical analysis, parsing, recursive - decent bottom - up parsing

**UNIT- II:**

**Data, data types, and basic statements:** Names, variables, binding, type checking, scope, scope rules, lifetime and garbage collection, primitive data types, strings, array types, associative arrays, record types, union types, pointers and references, Arithmetic expressions, overloaded operators, type conversions, relational and boolean expressions , assignment statements , mixed mode assignments, control structures – selection, iterations, branching, guarded Statements

**UNIT- III:**

**Subprograms and implementations:** Subprograms, design issues, local referencing, parameter passing, overloaded methods, generic methods, design issues for functions, semantics of call and return, implementing simple subprograms, stack and dynamic local variables, nested subprograms, blocks, dynamic scoping

**UNIT- IV:**

**Object- orientation, concurrency, and event handling:** Object – orientation, design issues for OOP languages, implementation of object, oriented constructs, concurrency, semaphores,
Monitors, message passing, threads, statement level concurrency, exception handling, event handling

**UNIT -V:**

**Functional programming languages:** Introduction to lambda calculus, fundamentals of functional programming languages, Programming with Scheme, –
Programming with ML,

**UNIT -VI:**

**Logic programming languages:** Introduction to logic and logic programming, – Programming with Prolog, multi - paradigm languages

**OUTCOMES:**

- Describe syntax and semantics of programming languages
- Explain data, data types, and basic statements of programming languages
- Design and implement subprogram constructs, Apply object - oriented, concurrency, and event handling programming constructs
- Develop programs in Scheme, ML, and Prolog
- Understand and adopt new programming languages

**TEXT BOOKS:**


1. Robert W. Sebesta, "Concepts of Programming Languages", Tenth Edition, Addison Wesley, 2012.
2. Programming Langugaes, Principles & Paradigms, 2ed, Allen B Tucker, Robert E Noonan, TMH


**REFERENCE BOOKS:**


1. R. Kent Dybvig, "The Scheme programming language", Fourth Edition, MIT Press, 2009.
2. Jeffrey D. Ullman, "Elements of ML programming", Second Edition, Prentice Hall, 1998.
3. Richard A. O'Keefe, "The craft of Prolog", MIT Press, 2009.
4. W. F. Clocksin and C. S. Mellish, "Programming in Prolog: Using the ISO Standard", Fifth Edition, Springer, 2003

**II Year – II Semester**

| L | T | P | C |
|---|---|---|---|
| 0 | 0 | 3 | 2 |

## ADVANCED DATA STRUCTURES LAB

**OBJECTIVES:**

- To understand heap and various tree structures like AVL, Red-black, B and Segment trees
- To understand the problems such as line segment intersection, convex shell and Voronoi diagram

**Programming:**

1. To perform various operations i.e., insertions and deletions on AVL trees.
2. To implement operations on binary heap.
   i) Vertex insertion
   ii) Vertex deletion
   iii) Finding vertex
   iv) Edge addition and deletion
3. To implement Prim's algorithm to generate a min-cost spanning tree.
4. To implement Krushkal's algorithm to generate a min-cost spanning tree.
5. To implement Dijkstra's algorithm to find shortest path in the graph.
6. To implementation of Static Hashing (Use Linear probing for collision resolution)
7. To implement of Huffmann coding.
8. To implement of B-tree.

**OUTCOMES:**

- Implement heap and various tree structure like AVL, Red-black, B and Segment trees
- Solve the problems such as line segment intersection, convex shell and Voronoi diagram

**II Year – II Semester**

| L | T | P | C |
|---|---|---|---|
| 0 | 0 | 3 | 2 |

## JAVA PROGRAMMING LAB

**Exercise - 1** (Basics)

a). Write a JAVA program to display default value of all primitive data type of JAVA

b). Write a java program that display the roots of a quadratic equation ax2+bx=0. Calculate the discriminate D and basing on value of D, describe the nature of root.

c). Five Bikers Compete in a race such that they drive at a constant speed which may or may not be the same as the other. To qualify the race, the speed of a racer must be more than the average speed of all 5 racers. Take as input the speed of each racer and print back the speed of qualifying racers.

d) Write a case study on **`public static void main`(250 words)**

**Exercise - 2** (Operations, Expressions, Control-flow, Strings)

a). Write a JAVA program to search for an element in a given list of elements using binary search mechanism.

b). Write a JAVA program to sort for an element in a given list of elements using bubble sort

(c). Write a JAVA program to sort for an element in a given list of elements using merge sort.

(d) Write a JAVA program using StringBufferto delete, remove character.

**Exercise - 3** (Class, Objects)

a). Write a JAVA program to implement class mechanism. – Create a class, methods and invoke them inside main method.

b). Write a JAVA program to implement constructor.

**Exercise - 4** (Methods)

a). Write a JAVA program to implement constructor overloading.

b). Write a JAVA program implement method overloading.

**Exercise - 5** (Inheritance)

a). Write a JAVA program to implement Single Inheritance

b). Write a JAVA program to implement multi level Inheritance

c). Write a java program for abstract class to find areas of different shapes

**Exercise - 6** (Inheritance - Continued)

a). Write a JAVA program give example for “super” keyword.

b). Write a JAVA program to implement Interface. What kind of Inheritance can be achieved?

**Exercise - 7** (Exception)

a).Write a JAVA program that describes exception handling mechanism

b).Write a JAVA program Illustrating Multiple catch clauses

**Exercise – 8 (**Runtime Polymorphism)

a). Write a JAVA program that implements Runtime polymorphism

b). Write a Case study on run time polymorphism, inheritance that implements in above problem

**Exercise – 9** (User defined Exception)

a). Write a JAVA program for creation of Illustrating throw

b). Write a JAVA program for creation of Illustrating finally

c). Write a JAVA program for creation of Java Built-in Exceptions

d).Write a JAVA program for creation of User Defined Exception

**Exercise – 10** (Threads)

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

b). Write a program illustrating **isAlive** and **join ()**

c). Write a Program illustrating Daemon Threads.

**Exercise - 11** (Threads continuity)

a).Write a JAVA program Producer Consumer Problem

b).Write a case study on thread Synchronization after solving the above producer consumer problem

**Exercise – 12** (Packages)

a). Write a JAVA program illustrate class path

b). Write a case study on including in class path in your os environment of your package.

c). Write a JAVA program that import and use the defined your package in the previous Problem

**Exercise - 13** (Applet)

a).Write a JAVA program to paint like paint brush in applet.

b) Write a JAVA program to display analog clock using Applet.

c). Write a JAVA program to create different shapes and fill colors using Applet.

**Exercise - 14** (Event Handling)

a).Write a JAVA program that display the x and y position of the cursor movement using

Mouse.

b).Write a JAVA program that identifies key-up key-down event user entering text in a Applet.

**Exercise - 15** (Swings)

a).Write a JAVA programto build a Calculator in Swings

b). Write a JAVA program to display the digital watch in swing tutorial.

**Exercise – 16 (Swings - Continued)**

a). Write a JAVA program that to create a single ball bouncing inside a JPanel.

b). Write a JAVA program JTree as displaying a real tree upside down

**III Year – I Semester**

| L | T | P | C |
|---|---|---|---|
| 4 | 0 | 0 | 3 |

## COMPILER DESIGN

**OBJECTIVES:**

- Understand the basic concept of compiler design, and its different phases which will be helpful to construct new tools like LEX, YACC, etc.

**UNIT – I**

Introduction Language Processing, Structure of a compiler the evaluation of Programming language, The Science of building a Compiler application of Compiler Technology. Programming Language Basics.

Lexical Analysis-: The role of lexical analysis buffing, specification of tokens. Recognitions of tokens the lexical analyzer generator lexical

**UNIT –II**

Syntax Analysis -: The Role of a parser, Context free Grammars Writing A grammar, top down passing bottom up parsing Introduction to Lr Parser.

**UNIT –III**

More Powerful LR parser (LR1, LALR) Using Armigers Grammars Equal Recovery in Lr parser Syntax Directed Transactions Definition, Evolution order of SDTS Application of SDTS. Syntax Directed Translation Schemes.

**UNIT – IV**

Intermediated Code: Generation Variants of Syntax trees 3 Address code, Types and Deceleration, Translation of Expressions, Type Checking. Canted Flow Back patching?

**UNIT – V**

Runtime Environments, Stack allocation of space, access to Non Local date on the stack Heap Management code generation – Issues in design of code generation the target Language Address in the target code Basic blocks and Flow graphs. A Simple Code generation.

**UNIT –VI**

Machine Independent Optimization. The principle sources of Optimization peep hole Optimization, Introduction to Date flow Analysis.

**OUTCOMES:**

- Acquire knowledge in different phases and passes of Compiler, and specifying different types of tokens by lexical analyzer, and also able to use the Compiler tools like LEX, YACC, etc.
- Parser and its types i.e. Top-down and Bottom-up parsers.
- Construction of LL, SLR, CLR and LALR parse table.
- Syntax directed translation, synthesized and inherited attributes.
- Techniques for code optimization.

**TEXT BOOKS:**


1. Compilers, Principles Techniques and Tools.Alfred V Aho, Monical S. Lam, Ravi Sethi Jeffery D. Ullman,2nd edition,pearson,2007

2. Compiler Design K.Muneeswaran, OXFORD

3. Principles of compiler design,2nd edition,Nandhini Prasad,Elsebier.


**REFERENCE BOOKS:**


1. Compiler Construction, Principles and practice, Kenneth C Louden, CENGAGE

2. Implementations of Compiler, A New approach to Compilers including the algebraic methods, Yunlinsu ,SPRINGER