II Year – II SEMESTER

T	P	C
4	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 - Concepts and Practices: Ugrasen Suman, Cengage Learning
2. Software Engineering - A Practitioner's Approach, Roger S. Pressman, Seventh Edition McGrawHill International Edition.
3. Fundamentals of Software Engineering, Rajib Mall, Third Edition, PHI.
4. 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, PankajJalote, 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++, Ellis Horowitz, Sartaj Sahni and Dinesh Mehta, 2nd Edition, Universities Press (India) Pvt. Ltd
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