5TH SEMESTER BSc. CS(H)			
	SUB CODE		SUB NAME
MAJOR	CORE-I	PAPER-11	Software Engineering
	CORE-I	PAPER-12	An Introⁿ to AI/ Computer Graphics
	CORE-I	PAPER-13	Programming in JAVA
MINOR	CORE-III	PAPER-3	Real Analysis-I
SEC	PAPER-2		
VAC	PAPER-3		

Semester V

Core XI **Software Engineering**

Course Objectives:

- To understand importance of Software engineering.
- To understand different software development models
- To understand various issues involved in a software development project

Learning Outcomes:

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

- Understand various software development lifecycle models
- Know the complexities involved in software development projects & how to deal with them
- Understand the software design process starting from requirement analysis
- Learn about software documentation, software testing and maintenance

Unit-I:

Introduction: Evolution of Software to an Engineering Discipline, Software Development Projects, Exploratory Style of Software Development, Emergence of Software Engineering, Changes in Software Development Practices, Computer Systems Engineering. Software Lifecycle Models: Waterfall Model and its Extensions, Rapid Application Development (RAD), Agile Development Models, Spiral Model.

Unit-II:

Software Project Management: Software Project Management Complexities, Responsibilities of a Software Project Manager, Project Planning, Metrics for Project Size Estimation, Project Estimation Techniques, Empirical Estimation Techniques, COCOMO, Halstead's Software Science, Staffing Level Estimation, Scheduling, Organization and Team Structures, Staffing, Risk Management, Software Configuration Management.

Unit-III:

Requirement Analysis and Specification: Requirements Gathering and Analysis, Software Requirement Specifications, Formal System Specification Axiomatic Specification, Algebraic Specification, Executable Specification and 4GL.

Software Design: Design Process, characterize a Good Software Design, Cohesion and Coupling, Layered Arrangements of Modules, Approaches to Software Design (Function Oriented & Object-Oriented).

Unit-IV:

Coding and Testing: Coding: Code Review, Software Documentation, Testing, Unit Testing, Black Box and White Box Testing, Debugging, Program Analysis Tools, Integration Testing, System Testing, Software Maintenance.

Text Book:

- ✓ *Fundamental of Software Engineering, Rajib Mall, Fifth Edition, PHI Publication, India.*

Reference Books:

- ✓ *Software Engineering– Ian Sommerville, 10/Ed, Pearson.*
- ✓ *Software Engineering Concepts and Practice – Ugrasen Suman, Cengage Learning India Pvt, Ltd.*
- ✓ *Software Engineering, R Khurana, Vikash Pubs.*

Core XI- Lab: Software Engineering

Students have to do at least two software development projects from the list of projects given below. They have to follow the complete software development lifecycle with the following details. UML can be used as a design tool. (Coding is optional).

1. • Problem Statement
   • Process Model
2. Requirement Analysis:
   - Creating a Data Flow
   - Data Dictionary, Use Cases
3. Project Management:
   - Computing FP
   - Effort
   - Schedule, Risk Table, Timeline chart
4. Design Engineering:
   - Architectural Design
   - Data Design, Component Level Design
5. Testing:
   - Basis Path Testing

List of Projects:

1. Criminal Record Management: Implement a criminal record management system for jailers, police officers and CBI officers.
2. Route Information: Online information about the bus routes and their frequency and fares
3. Car Pooling: To maintain a web-based intranet application that enables the corporate

employees within an organization to avail the facility of carpooling effectively.

4. Patient Appointment and Prescription Management System
5. Organized Retail Shopping Management Software
6. Online Hotel Reservation Service System
7. Examination and Result computation system
8. Automatic Internal Assessment System
9. Parking Allocation System
10. Wholesale Management System

Core XII (A): Introduction to Artificial Intelligence

Course Objectives:

- To learn the basic concepts of AI.
- To understand AI problem-solving approaches

Learning Outcomes:

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

1. Understand state space search as an approach to AI problem solving
2. Understand various Knowledge Representation techniques
3. Learn the complexity involved in NLP & role of learning in AI problem-solving
4. Understand the importance of Expert systems and the use of AI programming languages.

Unit-I:
Introduction to AI, Scope of AI, Characteristics of AI problems, Turing test, Concept of Intelligent agents, Approaches to AI problem-solving, State space search, production system, Uninformed search: Breadth-First, Depth-First, Iterative deepening, bidirectional and beam search.

UNIT-2:
Informed/Heuristic search: Generate-and-Test, Hill climbing, Best-first search, A^* algorithm, Problem reduction, AO^*, Constraint satisfaction, Solution of CSP using search, Means-End analysis.

UNIT-3:
Knowledge Representation: Propositional logic and Predicate logic along with their resolution principles, Unification algorithm, forward and backward chaining and conflict resolution, Semantic nets, Frames, Conceptual dependencies, Scripts.
Reasoning under uncertainty: Bayesian Belief networks, Dempster Shafer theory

UNIT-4:
Natural language processing: Introduction, Levels of knowledge in language understanding, , Phases of Natural language understanding, top-down and bottom-up parsing, transition networks.
Expert Systems: Introduction, Architecture, Expert system development cycle, Examples of ES: Mycin and Dendral.

Text Books:

- ✓ *Artificial Intelligence - A Modern Approach by Stuart J. Russell & Peter Norvig, Prentice Hall*
- ✓ *Artificial Intelligence by Rajiv Chopra, S. Chand Pubs.*

Reference Books:

- ✓ *D.W. Patterson, Introduction to A.I and Expert Systems, PHI Pub.*
- ✓ *Artificial Intelligence by Rich, Knight, and Nair, McGraw Hill*

Core XII (A) -Lab: Artificial Intelligence

1. Write a Python program to implement Depth-First Search (DFS) for a given graph. Test your program on a graph with at least 5 nodes. Verify your program by printing the order in which nodes are visited.
2. Write a Python program to implement Breadth-First Search (BFS) for a given graph. Use a queue to manage the nodes to be explored. Test your program on a graph with at least 5 nodes and print the order of node visits
3. Write a Python program to implement Uniform Cost Search (UCS) for finding the shortest path in a weighted graph. Test your program on a graph with at least 5 nodes and varying edge weights.
4. Write a Python program to implement the A* search algorithm. Your program should take a graph, a start node, a goal node, and a heuristic program as input. Test your implementation on a grid-based graph where the heuristic is the Manhattan distance.
5. Write a Python program to implement Greedy Best-First Search. Use a heuristic program to guide the search.
6. Write a Python program to solve a maze using the A* search algorithm. Represent the maze as a grid, where 0 indicates an open cell and 1 indicates a wall. Use Manhattan distance as the heuristic.
7. Write a Python program to implement the Minimax algorithm with Alpha-Beta pruning for a simple game (e.g., Tic-Tac-Toe).
8. Write a Python program to implement the Hill Climbing algorithm with random restarts. Test your program on a problem where the solution landscape has multiple peaks.
9. Write a Python program to represent the state of the 8-puzzle. Use a 2D list or a single list with 9 elements to represent the tiles. Implement a program to display the puzzle state.
10. Write a Python program to generate all possible moves (up, down, left, right) from a given state in the 8-puzzle. Ensure that your program checks for the boundaries of the puzzle.

Core XII (B) : Computer Graphics

Course Objectives:

- To understand basic concepts of computer graphics.
- To learn techniques for creating basic graphical structures
- To learn different transformation techniques

Learning Outcomes:

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

- Know the use of different graphics systems
- Learn different algorithms to draw geometrical figures
- Learn various geometric transformation techniques
- Learn techniques for clipping

Unit-I:

Computer Graphics: A Survey of Computer graphics, Overview of Graphics System: Video Display Devices, Raster-Scan Systems, Input Devices, Hard-Copy Devices, Graphics Software.

Unit-II:

Graphics Output Primitives: Point and Lines, Algorithms for line, circle & ellipse generation, Filled-Area Primitives. Attributes of Graphics Primitives: Point, line, curve attributes, fill area attributes, Fill methods for areas with irregular boundaries.

Unit-III:

Geometric Transformations (both 2-D & 3-D): Basic Geometric Transformations, Transformation Matrix, Types of transformation in 2-D and 3-D Graphics: Scaling, Reflection, shear transformation, rotation, translation. 2-D, 3-D transformation using homogeneous coordinates.

Unit-IV:

Two-Dimensional Viewing: Introduction to viewing and clipping, viewing transformation in 2-D, viewing pipeline, Clipping Window, Clipping Algorithms: Point clipping, Line clipping and Polygon clipping.

Text Books:

- ✓ *Donald Hearn & M. Pauline Baker, "Computer Graphics with OpenGL", Pearson Education.*
- ✓ *Mathematical Elements for Computer Graphics, D. F. Rogers & J. A. Adams, MGH, 2/ed.*

Reference Books:

- ✓ *Computer Graphics principles & practice, Foley, Van Dam, Feiner, Hughes Pearson Education*
- ✓ *Computer Graphics by Zhigang Xiang, Roy A Plastic, McGraw-Hill*

Core XII (B) - Lab: Computer Graphics using OpenGL

1. Write a program to implement Bresenham's line drawing algorithm.
2. Write a program to implement mid-point circle drawing algorithm.
3. Write a program to clip a line using Cohen and Sutherland line clipping algorithm.
4. Write a program to clip a polygon using Sutherland Hodgeman algorithm.
5. Write a program to fill a polygon using Scan line fill algorithm.
6. Write a program to apply various 2D translation transformation.
7. Write a program to apply 2D object homogenous coordinates translation.
8. Write a program to apply various 2D rotation transformation.
9. Write a program to apply 2D object homogenous coordinates rotation.
10. Write a program to apply various 2D scaling transformation.
11. Write a program to apply 2D object homogenous coordinates scaling transformation.
12. Write a program to apply various 3D transformations on a 3D object and then apply parallel and perspective projection on it.

Core-XIII Programming in Java

Course Objectives:

- To learn Java for writing object-oriented programs
- To understand the use of different Java programming constructs
- To learn exception handling in Java and use of threads.

Learning Outcomes:
Upon completion of this course, students will be able to:

- Learn the basics of Java programming
- Create classes/objects and implement different forms of inheritance
- Use arrays and files in Java
- Learn about exception handling

Unit-I:

Introduction to Java: Java History, Architecture and Features, Understanding the semantic and syntax differences between C++ and Java, Compiling and Executing a Java Program, Variables, Constants, Keywords (super, this, final, abstract, static, extends, implements, interface) , Data Types, Wrapper class, Operators (Arithmetic, Logical and Bitwise) and Expressions, Comments, Doing Basic Program Output, Decision Making Constructs (conditional statements and loops) and Nesting, Java Methods (Defining, Scope, Passing and Returning Arguments, Type Conversion and Type and Checking, Built-in Java Class Methods). Input through keyboard using Command line Argument, the Scanner class, BufferedReader class.

Unit-II:

Object-Oriented Programming Overview: Principles of Object-Oriented Programming, Defining & Using Classes, Class Variables & Methods, Objects, Object reference, Objects as parameters, final classes, Garbage Collection. Constructor- types of constructors, this keyword, super keyword. Method overloading and Constructor overloading. Aggregation vs Inheritance, Inheritance: extends vs implements, types of Inheritance, Interface, Up-Casting, Down-Casting, Auto-Boxing, Enumerations, Polymorphism, Method Overriding and restrictions. Package: Pre-defined packages and Custom packages.

Unit-III:

Arrays: Creating & Using Arrays (1D, 2D, 3D and Jagged Array), Array of Object, Referencing Arrays Dynamically. Strings and I/O: Java Strings: The Java String class, Creating & Using String Objects, Manipulating Strings, String Immutability& Equality, Passing Strings To & From Methods, StringBuffer Classes and StringBuilder Classes. IO package: Understanding StreamsFile class and its methods, Creating, Reading, Writing using classes: Byte and

Character streams, FileOutputStream, FileInputStream, FileWriter, FileReader, InputStreamReader, PrintStream, PrintWriter. Compressing and Uncompressing File.

Unit-IV:

Exception Handling, Threading, Networking and Database Connectivity: Exception types, uncaught exceptions, throw, built-in exceptions, Creating your own exceptions; Multi-threading: The Thread class and Runnable interface, creating single and multiple threads, Thread prioritization, synchronization and communication, suspending/resuming threads. Using java.net package, Overview of TCP/IP and Datagram programming. Accessing and manipulating databases using JDBC.

Text Book:

E. Balagurusamy, "Programming with Java", TMH, 4/Ed

Reference Book:

Herbert Schildt, "The Complete Reference to Java", TMH, 10/Ed.

Core XIII- Lab: Programming in Java

1. To find the sum of any number of integers entered as command line arguments.
2. To find the factorial of a given number.
3. To convert a decimal to binary number.
4. To check if a number is prime or not, by taking the number as input from the keyboard.
5. To find the sum of any number of integers interactively, i.e., entering every number from the keyboard, whereas the total number of integers is given as a command line argument.
6. Write a program that show working of different functions of String and StringBufferclasss like setCharAt( ), setLength( ), append( ), insert( ), concat( )and equals( ).
7. Write a program to create a – "distance" class with methods where distance is computed in terms of feet and inches, how to create objects of a class and to see the use of this pointer
8. Modify the – "distance" class by creating constructor for assigning values (feetandinches) to the distance object. Create another object and assign second object as reference variable to another object reference variable. Further create a third object which is a clone of the first object.
9. Write a program to show that during function overloading, if no matching argument is found, then Java will apply automatic type conversions (from lower to higher data type).
10. Write a program to show the difference between public and private access specifiers. The program should also show that primitive data types are passed by value and objects are passed by reference and to learn use of final keyword.

11. Write a program to show the use of static functions and to pass variable length arguments in a function.
12. Write a program to demonstrate the concept of boxing and unboxing.
13. Create a multi-file program where in one file a string message is taken as input from the user and the function to display the message on the screen is given in another file (make use of Scanner package in this program).
14. Write a program to create a multilevel package and also creates a reusable class to generate Fibonacci series, where the function to generate Fibonacci series is given in a different file belonging to the same package.
15. Write a program that creates illustrates different levels of protection in classes/subclasses belonging to same package or different packages
16. Write a program – "DivideByZero" that takes two numbers a and b as input, computes a/b, and invokes Arithmetic Exception to generate a message when the denominator is zero.
17. Write a program to show the use of nested try statements that emphasizes the sequence of checking for catch handler statements.
18. Write a program to create your own exception types to handle situation specific to your application (Hint: Define a subclass of Exception which itself is a subclass of Throwable).
19. Write a program to demonstrate priorities among multiple threads.
20. Write a program to demonstrate different mouse handling events like mouseClicked( ), mouseEntered( ), mouseExited( ), mousePressed( ), mouseReleased( ) & mouseDragged( ).
21. Write a program to demonstrate different keyboard handling events.