NORTH EASTERN HILL UNIVERSITY SHILLONG

Four years Undergraduate Programme of B.Sc. Computer Science

(as per NEP 2020 guidelines)

Date of Approval by AC: 30th May and 21st June 2024

Preface

The 4-year BSc program in Computer Science is a comprehensive course of study that provides students with a solid foundation in the principles and practices of computer science. This program is typically designed to equip students with the knowledge and skills necessary to pursue careers in various computing fields or to continue their education at the graduate level. The program usually starts with foundational courses that cover essential concepts such as programming, algorithms, data structures, computer organization, and discrete mathematics. These courses establish the fundamental building blocks of computer science.As students' progress through the program, they delve deeper into core computer science areas. These courses include topics such as software engineering, databases, operating systems, computer networks, theory of computation, and artificial intelligence. Students gain a comprehensive understanding of these areas and develop proficiency in problem-solving, programming languages, and software development methodologies. Many courses incorporate project-based learning, where students work on real-world problems or create software applications under the guidance of faculty members. This hands-on experience helps students apply their knowledge, develop problem-solving skills, and gain practical exposure to the field of computer science.The students are provided with the opportunity to acquire valuable industry experience by working in tech companies, research labs, or other relevant organizations through internship programs. This help students apply theoretical knowledge in real-world settings, develop professional skills, and make industry connections.

Overall, the 4-year BSc program in Computer Science provides students with a comprehensive education in computer science principles, theories, and practical skills. It prepares them for a wide range of career opportunities in software development, data analysis, system administration, cyber security, research, and more.

Programme Outcomes (POs):

The expected outcome of the programme are-

1. Students will be able to design, develop, and implement software solutions usingappropriate algorithms, programming languages, and software engineering principles.
2. Students will have a strong foundation in computer science theory, including datastructures, algorithms, automata theory, and computer organization.
3. Students will be able to analyze and evaluate complex computer systems, identifyingstrengths, weaknesses, and potential improvements.

4. Students will have experience working in teams on software projects, includingcollaboration, project planning, and communication.
5. Students will be able to apply ethical principles to their work in computer science, including issues related to privacy, security, and intellectual property.
6. Students will be able to effectively communicate technical information to both technicaland non-technical audiences, including through technical reports and presentations.
7. Students will be able to adapt to new technologies and programming languages as theyemerge, and continue to learn and develop professionally throughout their careers.

Course Structure

Semester 1

Sub Code	Subject Name	Theory	Practical	Total	Contact Hours
CSC-100	Programming in C and Introduction to Data Structures (Major)	3	1	4	75
CSC-100	Programming in C and Introduction to Data Structures (Minor)	3	1	4	75
MDC-110.....119	Any of the available course as notified by the University from time to time.	3	-	3	45
AEC-120….129	Any of the available course as notified by the University from time to time.	3	-	3	45
SEC-130…...139	Any of the available course as notified by the University from time to time.	-	3	3	90
VAC-140	Any of the available course as notified by the University from time to time.	3	-	3	45
Total				20	

Semester 2

Sub Code	Subject Name	Theory	Practical	Total	Contact Hours
CSC-150	Database Management System (Major)	3	1	4	75
CSC-150	Database Management System (Minor)	3	1	4	75
MDC-160…..169	Any of the available course as notified by the University from time to time.	3	-	3	45
AEC-170…..179	Any of the available course as notified by the University from time to time.	3	-	3	45
SEC-180……189	Any of the available course as notified by the University from time to time.	-	3	3	90
VAC-190…...199	Any of the available course as notified by the University from time to time.	3	-	3	45
Total				20	

Semester 3

Sub Code	Subject Name	Theory	Practical	Total	Contact Hours
CSC-200	Object Oriented Programming Using Java	3	1	4	75
CSC-201	Digital Logic Design	4	-	4	60
MDC-210…..219	Any of the available course as notified by the University from time to time.	3	-	3	45
AEC-220…..229	Any of the available course as notified by the University from time to time.	2	-	2	30
SEC-230……239	Any of the available course as notified by the University from time to time.			3	45-90
VTC - 240…..249	Any of the available course as notified by the University from time to time.	1	3	4	105
Total				20	

Semester 4

Sub Code	Subject Name	Theory	Practical	Total	Contact Hours
CSC-250	Internet and Web Applications (**Major**)	3	1	4	75
CSC-251	Data Communication and Computer Networks (**Major**)	4	-	4	60
CSC-252	Operating System (**Major**)	3	1	4	75
CSC-253	Computer System Architecture (**Major**)	4	-	4	60
VTC-260….269	Any of the available course as notified by the University from time to time.	1	3	4	105
Total				20	

Semester 5

Sub Code	Subject Name	Theory	Practical	Total	Contact Hours
CSC-300	Software Engineering	4	-	4	60
CSC-301	Mathematical Foundations of Computer Science	4	-	4	60
CSC-302	Programming in Python	3	1	4	75
CSC-302	Programming in Python (**Minor**)	3	1	4	75
#CSC-303	Internship	-	4	4	120
Total				20	

Semester 6

Sub Code	Subject Name	Theory	Practical	Total	Contact Hours
CSC-350	Design and Analysis of Algorithms	4	-	4	60
CSC-351	Artificial Intelligence	4	-	4	60
CSC-352	Cryptography and Network Security	4	-	4	60
CSC-353	Machine Learning	3	1	4	75
VTC-360…369	Any of the available course as notified by the University from time to time	1	3	4	105
Total				20	

*MDC=Multidisciplinary course; AEC=Ability enhancement course; SEC=Skill enhancement course; VAC=Value added course, VTC=Vocational Education and Training course.

#Software Engineering techniques to be applied at internship.

CSC-200: Object Oriented Programming Using Java

(Contact Hours: 75, Credits: 3 + 1=4)

(Marks: 56(T)+19(P)=75 (External) + 25 (Internal)=100)

Course Objectives :

- CO1: Understand the basic syntax and concepts of Java programming language
- CO2: Apply object-oriented programming principles effectively.
- CO3: Develop proficiency in handling file I/O operations.
- CO4: Implement multithreading and concurrency concepts in Java applications.
- CO5: Work with programs that use a RDBMS for data storage.
- CO6: Understand and implement networking concepts through java programs.
- CO7: Design and develop GUI applications using Java Swing.

Learning Outcomes:

- LO1: Remembering: Recognize Java syntax, keywords, and fundamental concepts.
- LO2: Understanding: Explain object-oriented programming principles and their application in Java.
- LO3: Applying: Implement Java programs to solve various computational problems using object-oriented techniques.
- LO4: Analyzing: Analyze and debug Java code for errors and inefficiencies, applying object-oriented design principles.
- LO5: Evaluating: Critique and optimize Java programs for performance and maintainability, considering object-oriented design principles.
- LO6: Creating: Design and develop multithreaded Java programs, networking programs and GUI based applications using object-oriented methodologies.

Outline of the Course

Unit	Topic	Minimum Class Hours	External Marks	Internal Marks
I	Introduction to Java programming	15	18	19
II	Inheritance, Packages, Interface, Exceptions and Multithreading	15	19	
III	File I/O, Networking, JDBC and GUI Programming	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

Detailed Syllabus

Unit I: Introduction to Java programming 15 Hours

Introduction to Java and object-oriented programming (OOP) concepts:Abstraction, Encapsulation, Polymorphism and Inheritance. Overview of Java programming environment: Java Development Kit (JDK), Java Virtual Machine (JVM), Bytecode.

Java syntax and data types:Java keywords, primitive types and literals. Variables: declaration, initialization, scope and lifetime. Type conversion, automatic type promotion. Operators: Arithmetic, Bitwise, Relational, Logical, Conditional operators and Operator precedence.

Control statements: Decision making using if, if-else-if, nested if, switch-case; Looping using while, do-while, for and for-each, local variable type inference in a for loop; break and continue statements. Arrays: declaration and initialization for one-dimensional and two-dimensional arrays.

Introduction to classes, methods and objects: General form of a class, declaring objects, assigning object reference variable, methods (adding methods to a class, returning a value, parameterized methods), constructors (types of constructors), overloading methods and constructors, this keyword, instance variable hiding, Using object as parameters (passing and returning objects), access control (default, public, private and protected), classes (static, final, nested and inner), static keyword used with class, method and variable, command-line arguments, program input using Scanner class (constructors, setting delimiters), Garbage collection (finalize() method).

String and Math class: The string constructor, special string operations, character extraction, string searching and comparison, data conversion using valueOf (), StringBuffer, Math methods (transcendental, exponential, rounding)

Unit II: Inheritance, Packages, Interface, Exceptions and Multithreading 15 Hours

Inheritance: Extending a class, member access and inheritance, using super, creating a multilevel hierarchy, constructors and inheritance, method overriding, dynamic method dispatch, using final with inheritance, abstract classes (properties, utility), the Object class.

Packages and Interface: Packages (Defining a package, classpath, Access protection and importing packages), Interfaces (Defining and implementing interfaces, variables in interface, extending interfaces, default interface methods, static methods, private interface methods)

Exception handling: Exception handling fundamentals (try - catch), Nested try statement, multiple catch clauses, throw, throws and finally, exception types (checked and unchecked), built-in exceptions, Throwable and Exception class, user defined exception subclasses.

Multithreaded Programming: Java thread model (thread priorities, synchronization and inter-thread communication), creating threads (extending Thread, implementing Runnable), main thread, thread methods (isAlive( ), join( )), suspending, resuming and stopping threads, deadlock.

Unit III: File I/O, Networking, JDBC and GUI Programming 15 Hours

Input/Output: Streams (Byte Streams and Character streams), Standard streams (System.in, System.out, System.err), InputStream, OutputStream, Reader, Writer, PrintWriter, Reading and writing console I/O, Reading and writing files, FileInputStream, FileOutputStream, PrintStream, FileReader, FileWriter, try-with-resources, File class.

Networking: Networking basics, InetAddress, Factory methods, URL, URLConnection, HttpURLConnection, Establishing a simple server using Stream Sockets, Establishing a simple client using Stream Sockets, Connectionless Client/Server Interaction with Datagrams.

Java database connectivity (JDBC): Introduction to JDBC; packages, classes and interfaces for JDBC (Connection, Statement, PreparedStatement, ResultSet), establishing database connections, inserting, editing, deleting and displaying records from a RDBMS.

GUI programming using Swing and Event Handling: Creating a Swing application; Components and Containers: JPanel, JFrame, JTextField, JLabel, Swing buttons(JButton, JCheckBox and JRadioButton), JList, JComboBox, JMenuBar, JMenu, and JMenuItem, layouts (FlowLayout, BorderLayout, GridLayout), Handling events: The Delegation Event Model, Event Classes, Sources of events, Event Listener Interfaces, Processing mouse events, Handling keyboard events, Adapter classes.

Unit IV: Practical 30 Hours

List of practical assignments. (Problems may not be restricted to this list)

1. Write a program to create a class called Box with a parameterized constructor, along with a method to calculate the volume of the box. Use the class to find the volume of two boxes whose height, width and depth are 10,20,30 and 20,30,40 respectively.
2. Define a class called stack that can hold 10 integer values, then initialize top of the stack with push and pop methods. Write a program to push the elements into the stack and pop out from the stack.
3. Write a Java program using a class to multiply two matrixes of 3*3 order. Allow the user to input the values through the keyboard.
4. Write a Java program to find factorial of positive integer using recursion.
5. Write a Java program to accept the command line arguments and display the arguments along with the positions.
6. Write a Java program to demonstrate method overriding where the program creates a superclass called figure that stores the dimensions of various two-dimensional objects. It also defines a method called area() that computes the area of an object. The program derives two subclasses from figure. The first is Rectangle and the second is Triangle. Each of these subclass overrides area() so that it returns the area of a rectangle and a triangle respectively.
7. Write a Java program to create a thread and start running it using runnable interface. Allow the thread to display a message five times with a gap of 500ms.
8. Write a Java program to demonstrate the synchronization of two threads using the synchronized statement.
9. Write a Java program to demonstrate interthread communication considering the producer and consumer problem. There must be two classes one for producer to produce data and another is consumer to consume data [Hint: Use wait() and nothing() to signal in both directions].
10.Write a Java program to copy the content of one file to another using java.io

11.Write a Java program to read integer values from a text file (stored as one value per line), sort the values using selection/bubble/insertion sort and then store the result in another text file.
12.Design a Calculator System using Java, The application should have all the digit buttons along with buttons for operations +,-,*,/ and =. There is a designated panel to show the current results. If a digital button is clicked, the number is displayed on the panel. If an operator button is clicked the operation is to be performed. You may assume the expression to be infix.
13.Write a program to input integers into an array and sort them using methods. Display the sorted numbers.
14.Write a socket based Java application program to create a connection between two machines such that whatever text one machine is sending to the other will be displayed at the latter′s screen and vice-versa
15.Develop an application using swing to display four push buttons and a text field. Each button displays an icon that represents the flag of a country. When a button is pressed, the name of the country is displayed in the text field.
16.Write a Java program that prints the addresses and names of the local machine and two well-known web sites.

Instructions to Paper Setter (Theory)

Questions should be set according to the following scheme:

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
Total	**6**	**3**

Instructions to Paper Setter (Practical)

Questions should be set according to the following scheme:

Unit	Questions to be set	Questions to be answered
IV (Practical)	Section I (from Unit I): 2	1
	Section II (from Unit II & III): 3	2

Exam Duration:

Theory	3 hours
Practical	3 hours

Suggested Readings

Text:

1. H.Schildt, *Java: The Complete Reference*, 13thedition, McGraw Hill, 2024.

References :

1. P. Deitel, H. Deitel, *Java How to Program*: *Early Objects*, 11th edition, Pearson Education, 2018.
2. D.Liang, *Intro To Java Programming, Comprehensive Version*, 10th edition, Pearson Education, 2018.
3. E. Balagurusamy, *Programming with Java*, 7thedition, McGraw Hill, 2023

CSC-201: Digital Logic Design
(Hours : 60, Credits : 4)

(Marks: 75 (External) + 25 (Internal)=100)

Course Objectives

- CO1:This undergraduate paper aims to equip students with a comprehensive understanding of core concepts and skills in Digital Logic Design.
- CO2: Students will learn to design circuits using basic logic gates and comprehend combinational and sequential logic principles.

Learning Outcomes

- LO1: Define and explain key digital logic concepts, including different number representations, Boolean algebra, and logic gates.
- LO2: Demonstrate the ability to perform basic operations in binary and understand their equivalents in Boolean algebra.
- LO3: Design and analyze simple digital circuits using fundamental logic gates and evaluate the functionality of designed circuits through truth tables and Boolean expressions.
- LO4: Apply simplification techniques to optimize and reduce logic circuits.
- LO5: Design and implement combinational logic circuits using multiple logic gates.
- LO6: Understand the concept of sequential logic and the role of flip-flops in digital circuits.
- LO7: Design sequential circuits and apply state diagrams and excitation tables to describe the behaviour of sequential circuits.

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Number Systems and Boolean Algebra	15	15	25
II	Boolean Functions and Simplification	15	20	
III	Combinational Logic Circuits	15	20	
IV	Sequential Logic Circuits	15	20	
Total		60	75	25

Detailed Syllabus

Unit I: Number Systems and Boolean Algebra 15 Hours

Binary Systems: Digital Computer, Binary Numbers, Number Base Conversions, Octal and Hexadecimal Numbers, Complements – r’s complement, (r-1)’s complement, Subtraction using r’s complement and (r-1)’s complement, Binary Code-BCD, Alphanumeric codes – ASCII code, Parity Bit, Storage and Registers – Introduction

Boolean Algebra and Logic Gates: Basic Definitions, Axiomatic Definition of Boolean Algebra, Basic Theorems of Boolean Algebra, Duality; Operator precedence; Digital Logic Gates - AND, OR, Inverter, Buffer, XOR, XNOR, NAND, NOR.

Unit II: Boolean Functions and Simplification **15 Hours**

Boolean Functions : Truth Table representation of a function, implementation of Boolean function using logic gates, algebraic manipulation, complement of a function; Canonical Forms (Minterm and Maxterm) and Standard Forms (Sum of Products representation of Boolean functions, Product of Sums representation of Boolean functions), Conversion between Cannonical forms, Standard Form.

Simplification of Boolean Functions: The Karnaugh Map Method- Two-variable map, Three-variable map, Four-variable map, Don't-care Conditions, Sum of Products Simplification, Product of Sums Simplification using NAND and NOR Implementation, The Tabulation Method, Determination of Prime-Implicants.

Unit III : Combinational Logic Circuits **15 Hours**

Combinational Logic: Introduction, Design Procedure, Adders- Half Adder, Full Adder; Subtractors – Half Subtractor, Full Subtractor and construction using Basic Logic Gates (OR, AND, NOT) and Universal Logic Gates (NAND & NOR);Binary Parallel Adder, Magnitude Comparator, Decoder, Encoder, Multiplexer.

Unit IV: Sequential Logic Circuits **15 Hours**

Sequential Logic: Introduction, Flip-Flops- RS Flip flop, D Flip Flop, JK Flip Flop; Sequential Circuit– State Diagram, Flip-flop Excitation Table, Flip Flop input-output equation, Design of Counters; Introduction to Register, Counter: Shift Register, Ripple Counter, Synchronous-counter-Binary counter, Binary Up-Down counter.

Instructions to Paper Setter

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
IV	2	1
Total	**8**	**4**

Exam Duration

Theory	3 Hours
Practical	-

Suggested Readings

Texts:

1. M. Morris Mano, *Digital Logic and Computer Design*, 1st Edition, Pearson Education, 2016

References:

1. M. Morris Mano, *Computer System Architecture*, 3rd Edition, Pearson Education, 2017
2. V. Carl Hamacher, Zvonko G. Vranesic, Safwat G. Zaky, *Computer Organization*, 5th Edition, Mc Graw Hill, 2002

CSC-250: Internet and Web Applications
(Hours: 75, Credits: 3+1=4)

(Marks: 56(T)+19(P)=75 (External) + 25 (Internal)=100)

Course Objectives

- CO1: Develop skills in web programming, including markup and scripting languages.
- CO2: Introduce structure and object-oriented programming design.
- CO3: Enhance understanding of the use of HTML5, JavaScript, and PHP in web development.
- CO4: Learn to create dynamic web pages that interact with users and process form data securely.

Learning Outcomes

- LO1: Develop skills in designing user interfaces (UI) using HTML5 and CSS
- LO2: Understand and apply basic programming syntaxes such as variables, data types, strings, expressions, and operators in JavaScript and PHP.
- LO3: Analyze and create dynamic web pages that utilize JavaScript and PHP for user interactions and form processing.
- LO4: Learn to handle user input, process form data, and create comprehensive forms using PHP for server-side scripting.
- LO5: Gain experience in connecting to databases via PHP for data storage and retrieval.
- LO6: Learn to implement projects that integrate HTML5, CSS, MySQL, and PHP to create functional web applications, such as an image microblogging app.
-

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Introduction to Web Technology	15	19	19
II	Client-side Scriptingusing Javascript	15	19	
III	Server-Side Programming with PHP	15	18	
IV	Practical	30	19	6
Total		75	75	25

Detailed Syllabus

UnitI: Introduction to Web Technology 15 Hours

Web Basics: Web Browsers, Web Servers, Tier Technologyand its types, Static and Dynamic Web Pages. Client-side and Server-side Scripting.

Web Protocols: details of HTTP, HTTPs, FTP

HTML5: Document metadata; Basic structure of HTML5; Sections; Grouping content;Text-level semantics;Embedded content;Tabular data;Forms;Interactive elements; List; Links; Images; *Page Designing with CSS:* Introduction to designing approaches;Table-based designs; Table-less designs;Cascading Style Sheet and its properties; Introduction; CSS vs CSS3; CSS properties — Text and Fonts, Colors and Backgrounds, The Box Model (dimensions, padding, margin and borders), Positioning and Display, Lists, Tables, Media.

UnitII: Client-Side Scripting using Javascript 15 Hours

Introduction to JavaScript: Lexical Structure; Variables, Identifiers, Data Types and Values, Scope, Literals, Reserved Words; Expression and operators, Statements; Arrays, Objects (Math, String, Date);Functions; Regular Expressions; Garbage Collection; Objects; Objects and properties; Constructors;Prototype and Inheritance; Object as an associative array; DOM and Event Handling

UnitIII: Server-Side Programming with PHP 15 Hours

Introduction to server-side programming: PHP Basics, Object-oriented Concepts; Embedding PHP script;Basic syntax (Variables, operators, expressions, constants); Control structures;PHP functions; Recursion; String manipulation; Using regular expression; Exceptional handling with PHP

*Database Connectivity in PHP:*Introduction to SQL;Basic SQL commands (CRUD); HTML forms and Methods; Database connectivity; MySQL functions;Executing DDL and DML queries using PHP; Login and authentication; Session and Cookies

UnitIV: Practical 30 Hours

List of practical assignments. (Problems may not be restricted to this list)

1. Create an HTML document for yourself, including your name, address, e-mail address, phone number, date of birth and age. If you are a student, you must include the course you have undertaken and describe a little bit about the course. If you are employed, you must include your employer, your employer's address and your job title. This document must use several headings and <em>, <strong>,<hr/>,<p> and <br/> tags.
2. Create an HTML document to describe an unordered list of a typical grocery shopping list you write.
3. Create an HTML document to describe an unordered list of at least four states. Each element of the list must have a nested list of at least three cities in the state.
4. Create an HTML document to describe an ordered list of your favourite top ten movies.
5. Modify the list of Exercise 4 to add nested, unordered lists of at least two actors and/or actresses in your favourite movies.
6. Create an HTML document to describe an ordered list with the following contents: the highest level should be the names of your two parents, with your mother first. Under each

parent, you must have a nested, ordered list with the brothers and sisters of your parents, in order with the eldest first. Each of the nested lists must have nested lists that list the children of your uncles and aunts (your cousins)-under the proper parents of course. Regardless of how many aunts, uncles and cousins you have, there must be at least three list items in each sublist below each of your parents and each of your aunts and uncles.

7. Create an HTML document that defines a table with columns for state, state bird/animal, state flower and state food. There must be at least five states as rows in the table.
8. Create an HTML document that has a form with the following controls:
    a. A text box to collect the user's name.
    b. Four checkboxes, one each for the following items:
        i. Four 100-watt light bulbs for Rs70.
        ii. Eight 100-watt light bulbs for Rs140.
        iii. Four 100-watt long-life light bulbs for Rs90.
        iv. Eight 100-watt long-life light bulbs for rs210
    c. A collection of three radio buttons that are labelled as follows:
        i. Visa
        ii. Mastercard
        iii. Maestro.
9. Create an HTML document that has six short paragraphs of text that describe various aspects of the state in which you live. You must define three paragraph styles p1,p2 and p3. The p1 style must use left and right margins of 20 pixels, a baackground color of pink and a foreground color of blue. The p2 style must use left and right margins of 30 pixels, a background colour of black and a foreground colour of yellow. The p3 style must use a text-indent of 1 cm, a background colour of green and a foreground colour of white. The first and fourth paragraphs must use p1, the second and fifth must use p2 and the third and sixth must use p3.
10. Create an HTML document that describes nested ordered lists of cars. The outer list must have three entries: compact, midsize and sports. Inside each of these three lists, there must be two sublists of body styles. The compact and midsize car sublists are two-door and four-door, and the sports car sublists are coupe and convertible. Each body-style sublist must have at least three entries, each of which is the make and model of a particular car that fits the category. The outer list must use uppercase Roman numerals, the middle lists must use uppercase letters and the inner list must use Arabic numerals. The background colour for the compact car list must be pink; for the midsize car list it must be blue; for the sports car list, it must be red. All of the styles must be in a document style sheet.
11. Create an HTML document that contains an unordered list of at least five popular books. The bullet for each book must be a small image of the book's cover.
12. Write a Javascript code to display a table of the numbers from 5 to 15 and their squares and cubes through an HTML document. [Hint: Use documentwrite to display output in a tabular form, using the assistance of table HTML tags]. Use for loop or do loop.
13. Write a Javascript code to display the first 50 Fibonacci Numbers through an HTML document. [Hint: Use document.write to display output in a tabular form, using the assistance of table HTML tags]. Use for loop or do loop.

14. Write a Javascript code to display a list of Armstrong numbers between 100 and 1000 through an HTML document. [Hint: Use document.write to display output in a tabular form, using the assistance of table HTML tags]. Use for loop or do loop.
15. Write a Javascript code to display a table of Palindrome numbers between 100 and 500 through an HTML document. [Hint: Use document.write to display output in a tabular form, using the assistance of table HTML tags]. Use for loop or do loop.
16. Write a Javascript code to generate a list of numbers between 100 and 1000 where the result of the current number is the sum of the previous four numbers in the series. For example,the initial four numbers are 0,1,1,2. The next number in the series should be 4.
17. Write a Javascript code to display a list of odd numbers from 1 to 100.
18. Write a Javascript code to display all prime numbers between 1 and 100.

19. A company wants to transmit data over the telephone, but it is concerned that its phones may
    be tapped. All of its data are transmitted as four-digit integers. It has asked you to write a program that will encrypt its data so that the data may be transmitted more securely. Your script should specify the input as 4561 and encrypt it as follows: Replace each digit by *(the sum of that digit plus 7) modulus 10*. Then swap the first digit with the third, and swap the second digit with the fourth. Then output HTML text that displays the encrypted integer.
20. Use a single-subscripted array to solve the following problem: A company pays its salespeopleon a commission basis. The salespeople receive $200 per week plus 9% of their gross sales forthat week. For example, a salesperson who grosses $5000 in sales in a week receives $200 plus 9of $5000 or a total of $650. Write a script (using an array of counters) that obtains the gross sales foreach employee through an HTML form and determines how many of the salespeople earned salariesin each of the following ranges (assume that each salesperson's salary is truncated to an integeramount):
    a) $200-$299
    b) $300-$399
    c) $400-$499
    d) $500-$599
    e) $600-$699
    f) $700-$799
    g) $800-$899
    h) $900-$999
    i) $1000 and over

21. Use a single-subscripted array to solve the following problem: Read in 20 numbers, each of which is between 10 and 100, inclusive. As each number is read, print it only if it is not a duplicate of a number that has already been read.
22. *(Airline Reservations System)* A small airline has just purchased a computer for its new automated program to assign seats on each flight of the airline's only plane (capacity: 10 seats). Your program should display the following menu of alternatives: **Please type 1 for "First Class"** and **Please type 2 for "Economy"**. If the person types **1**, your program should assign a seat in the first-class section (seats 1–5). If the person types **2**, your program should assign a seat in the economy section (seats 6–10). Your program should print a boarding pass indicating the person's seat number and whether it is in the first-class or economy section of the plane. Use a single-subscripted array to represent the seating chart of the plane. Initialize all the elements of the array to **0** to indicate that all seats are empty. As each seat is assigned, set the corresponding elements of the array to **1** to indicate that the seat is no longer available. Your program should, of course, never assign a seat that has

already been assigned. When the first-class section is full, your program should ask the person if it is acceptable to be placed in the economy section (and vice versa). If yes, then make the appropriate seat assignment. If no, then print the message **"Next flight leaves in 3 hours."**

23. A parking garage charges a $2.00 minimum fee to park for up to three hours. The garage charges an additional $0.50 per hour for each hour *or part thereof* in excess of three hours. The maximum charge for any given 24-hour period is $10.00. Assume that no car parks for longer than 24 hours at a time. Write a script that calculates and displays the parking charges for each customer who parked a car in this garage yesterday. You should input from the user the hours parked for each customer. The program should display the charge for the current customer and should calculate and display the running total of yesterday's receipts. The program should use the function **calculateCharges**to determine the charge for each customer.
24. Use Javascript to write a function **displayDigits**that receives an integer between **1** and **99999** and prints it as a series of digits, each pair of which is separated by two spaces. For example, the integer **4562** should be printed as **4 5 6 2**.
25. One interesting application of computers is the drawing of graphs and bar charts (sometimes called *histograms*). Write a script that reads five numbers between 1 and 30. For each number read, output XHTML text that displays a line containing that number of adjacent asterisks. For example, if your program reads the number 7, it should output XHTML text that displays *******.
26. Write a script that prints the following diamond shape:

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

27. Write a Javascript code to find the PigLatin form of any given string as input. **A PigLatin form of a word is extracting the first letter of the word and adding it to the end of that word and then adding a letter 'a'.** For example, if the word is **Hell**o, then the PigLatin form is **Elloha**.

Another example is if the input is **"Hello World"**, the output should be **"EllohaOrdlwa"**.

28. Write a Javascript code that creates customer bills for a carpet company when the following information is given as input:
    The length and width of the carpet in feet.
    The carpet price per square foot.
    The percentage of discount for each customer.
    The labour cost is fixed at Rs.10 per square foot and the tax rate is 8.5%. The bill should display the amounts under various heads like carpet cost, labour cost, discount, tax and total amount.
    Total amount=carpet cost+labor cost – discount + tax.
    Use arrays for inputting the number of customers whose bills are to be calculated.
29. An integer number is said to be a ***perfect number***if its factors, including 1 (but not the number itself), sum to the number. For example, 6 is a perfect number, because 6 = 1 + 2 + 3. Write a Javascript code that determines and displays all the perfect numbers between 1 and 1000. Print the factors of each perfect number to confirm that the number is indeed perfect.

30. Develop an XHTML document embedded with Javascript code that manages a one-dimensional array whose size is to be entered by the user. The program checks if the sum of the values of any two adjacent elements in the array is equal to a given number (inputted by the user) and displays the two elements of the array.

31. English ordinals follow a predictable, if not beautifully simple, set of rules:
    - ❑"st" is appended to 1 and numbers that are one greater than a multiple often, except for 11 and numbers that are 11 greater than a multiple of 100.
    - ❑"nd" is appended to 2 and numbers that are two greater than a multiple often, except for 12 and numbers that are 12 greater than a multiple of 100.
    - ❑"rd" is appended to 3 and numbers that are three greater than a multiple often, except for 13 and numbers that are 13 greater than a multiple of 100.
    - ❑"th" is appended to everything else.
32. The document must have a paragraph of at least 10 lines of text that describe you. This paragraph must be centred on the page and have space for 20 characters per line only. An image must be superimposed over the centre of the text as a nested element.
33. The document must have a paragraph of text that describes your home. Choose at least three different phrases (three to six words) from this paragraph and make them change the font, font style, colour and font size when the mouse cursor is placed over them. Each of the different phrases must change to different fonts, font styles, colours and font sizes.
34. Write a PHP program that tests whether an e-mail address is input correctly. Verify that the input begins with a series of characters, followed by the @ character, another series of characters, a period (.) and a final series of characters. Test your program, using both valid and invalid email addresses.
35. Write a PHP program that obtains a URL and a description of that URL from a user and stores the information in a database using MySQL. The database should be named URLs, and the table should be named Urltable. The first field of the database, which is named URL, should contain an actual URL, and the second, which is named Description, should contain a description of that URL. Use www.deitel.com as the first URL, and input Cool site! as its description. The second URL should be www.php.net, and the description should be The official PHP site. After each new URL is submitted, print the complete results of the database in a table.
36. Create a Database in a Database Server with two tables –Biodata and Marks. The table Biodata contains the fields Name, RollNo(N,5)(RollNo is unique), Gender(C,1), State(C,15), District(C,15), Place(C,15), Class(C,3),Dob(Date), Caste(C,10) and the table Marks contains the filedsRollNo(N,5), Physics(N,2), Chemistry(N,2), Maths(N,2). Develop PHP page(s) to Add, Edit and Delete records from the table. Provisions should be made to display all the records of the given class with each one's average mark in a tabular format (Class can be selected from a Pull Down Menu).

Instructions to Paper Setter (Theory)

Questions should be set according to the following scheme.

Unit	**Questions to be set**	**Questions to be answered**
I	2	1
II	2	1
III	2	1
Total	**6**	**3**

Instructions to Paper Setter (Practical)

Questions should be set according to the following scheme.

Unit	**Questions to be set**	**Questions to be answered**	**Marks**
I	3	2	6+6=12
II			
III	2	1	7
Total	**5**	**3**	**19**

Exam Duration

Theory	3 Hours
Practical	3 Hours

Suggested Readings

Texts:

1. DT Editorial Services, *HTML 5 Black Book,* 2nd Edition, Dreamtech Press, 2016
2. M H. Deitel, P. J**.** Deitel**,** *Internet and World Wide Web: How to Program,* 5th Edition, Pearson Education India, 2018
3. Ben Frain, *Responsive Web Design with HTML5 and CSS,* 3rd Edition, Packt Publishing, 2020

References:

1. Robert W. Sebesta, *Programming with World Wide Web,* 4th Edition, Pearson Education, 2011
2. UttamK.Roy, *Web Technologies*, 1st edition, Oxford University Press, 2010

CSC-251: Data Communication and Computer Networks
(Contact Hours: 60, Credits: 4)

(Marks: 75 (External) + 25 (Internal)=100)

Course Objectives

- CO1: Understand the fundamental concepts and principles of data communication and networking, including protocols, architectures, and technologies used in modern computer networks.
- CO2: Explore advanced topics in data communication and networking, including wireless communication, network management, and emerging technologies, to prepare students for real-world challenges in the field.

LearningOutcomes

After completing the course, students will be able:

- LO1: To develop an understanding of computer networks and communication basics
- LO2: Understand data transmission and media characteristics
- LO3: Understand the basic techniques of error detection and correction

OutlineoftheCourse

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Introduction to Computer Networks, Physical and Data Link Layer	15	18	25
II	Data Link Layer	15	19	
III	Network Layer	15	19	
IV	Transport Layer and Application Layer	15	19	
Total		60	75	25

Detailed Syllabus

Unit I: Introduction to Computer Networks, Physical and Data Link Layer 15 Hours

Uses of computer Networks: Business Applications, Home Applications, Mobile users; Network topology – Linear, Bus, Ring, Star, and Hybrid Topology; Network Hardware – LAN, MAN, WAN; Network Software- protocol Hierarchies, Connection Oriented and connectionless services; Reference models-OSI Reference model, TCP/IP Reference model.

Guided Transmission Media: Twisted pair, Fibre Optics; Wireless transmission- Radio Transmission, Microwave Transmission, Frequency Division and Time division Multiplexing; Switching-circuit switching, message switching, packet switching;

Unit II: Data Link Layer **15 Hours**

Services provided to the Network layer, Framing, Error control, Flow control, Error-detecting codes, Error-correcting code: Hamming codes; Simplex stop-and-wait protocol for an error-free channel; A simplex protocol for a noisy channel; Sliding window protocols (concepts only) - A One-Bit Sliding Window Protocol, A Protocol Using Go-Back-N, A go Back N and selective repeat protocols.

Unit III: Network Layer **15 Hours**

Design Issues – Store-and-forward Packet Switching, Implementation of Connectionless Service, Implementation of Connection-Oriented Service; Routing Algorithms- Shortest Path Routing, Flooding, Distance Vector Routing, Link State Routing, Hierarchical Routing, Routing for Mobile Hosts, Routing in Ad Hoc Networks; Congestion Control– Approaches to Congestion Control, Admission Control, Choke Packets, Explicit Congestion Notification, Hop-by-Hop Backpressure.

Unit IV: Transport and Application Layer **15 Hours**

Transport Service Primitives; Elements of Transport Protocols – Addressing, Connection Establishment, Connection Release, Flow Control and Buffering, Crash Recovery.

Internet Transport Protocols – UDP – Remote Procedure Call; TCP – TCP Service Model, Protocol, Header, Connection establishment, Connection Release, Transmission Policy.

Domain Name System - DNS Name Space, Domain Resource Records, Name Servers, Electronic Mail – Architecture and Services, User Agent, Message Transfer – SMTP, Message Delivery – POP3 and IMAP.

Instructions to Paper Setter

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
IV	2	1
Total	**8**	**4**

Exam Duration

Theory	3 Hours
Practical	-

Suggested Readings:

Text:

1. Andrew S. Tanenbaum, *Computer Networks*, 5th Edition, Pearson, 2010

References:

1. Sudakshina, Kundu, *Fundamentals of Computer Networks*, 2nd Edition, Prentice Hall of India , 2005
2. Behrouz A. Forouzan, *Data Communications and Networking*, 6th Edition, Tata McGraw Hill Edition, 2022

CSC-252: Operating System

(Contact Hours: 75, Credits: 3+1=4)

(Marks: 56(T)+19(P)=75 (External) + 25 (Internal)=100)

Course Objective

- CO1: The course is designed to provide students with a comprehensive understanding of the fundamental concepts, principles, and mechanisms underlying operating systems' design, implementation, and management.
- CO2: The aim is to equip students with the knowledge of operating system concepts while emphasizing practical skills and hands-on experience with Linux, considering the diverse needs of modern computing environments.

Learning Outcome

On successful completion of the course, students will be able to:

- LO1: Gain an understanding of fundamental operating system concepts such as process management, memory management, file systems, and I/O systems.
- LO2: Analyse and evaluate the performance of various algorithms such as process scheduling, memory management and disk management.
- LO3: Apply practical skills in using open source operating system like Linux, including navigating the command line interface, managing files and directories, and executing basic commands.
- LO4: Create shell scripts to automate tasks and manipulate files and directories efficiently.

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Concepts & Processes	15	19	19
II	Deadlocks & Memory Management	15	19	
III	I/O Management and File System Management	15	18	
IV	Practical	30	19	6
Total		75	75	25

Detailed Syllabus

UNITI: Concepts & Processes **15 Hours**

Operating system as a Resource Manager, History and features of various existing operating systems (Windows, macOS, Linux), the Operating System Zoo (Mainframe, Server, Multiprocessor, Personal Computer, Handheld, Embedded, Sensor-Node, Real-Time, Smart Card operating systems), Introduction to Processes (The Process Model, Process Creation, Process Termination, Process Hierarchies, Process States, Implementation of Processes, Process Control Block), Inter-process Communication (Race conditions, Critical Sections, Mutual Exclusion with Busy Waiting (Disabling interrupts, Lock variables, Peterson's solution), Sleep and wakeup, Semaphores, Mutexes (Introduction to Mutex, Futex), Message Passing, The Dining Philosophers Problem, Process Scheduling in batch systems (First Come First Served, Shortest Job First), Process scheduling in interactive systems(Round robin scheduling, Priority scheduling, Multiple queues).

UNITII: Deadlocks & Memory Management 15 Hours

Introduction, Deadlock (Conditions for Deadlock, Deadlock modelling), Deadlock detection and recovery (Deadlock detection with one resource of each type, Recovery from deadlock), Deadlock avoidance (Resource trajectories, safe and unsafe states, Bankers algorithm for single resource of each type, Bankers algorithm for multiple resource of each type), The four approaches of Deadlock Prevention, Memory management (No abstraction, Memory abstraction - the notion of an address space, Swapping, Managing free memory), Virtual Memory (Paging, Page Tables), Page Replacement Algorithms (The Not Recently Used, the First-In First-Out(FIFO), the Second-Chance Page Replacement Algorithm, The Least Recently Used (LRU) Page Replacement Algorithm), Design issues for Paging Systems(Page Size, Separate Instruction and Data Spaces, Shared Pages), Implementation issues (Operating System Involvement with Paging, Page Fault Handling, Locking Pages in Memory), Segmentation (Implementation of Pure Segmentation, Segmentation with Paging: MULTICS).

UNIT III: I/O Management and File System Management (Theory) 15 Hours

Principles of I/O hardware (I/O devices, Device Controllers, Direct memory access), Principles of I/O software(Programmed I/O, Interrupt-drive I/O, I/O using DMA), I/O Software Layers (Interrupt Handlers, Device Drivers, Device-Independent I/O Software, User-Space I/O Software), Disk hardware (Magnetic disks, RAID), Disk formatting, Disk Arm Scheduling Algorithms (FCFS, Shortest seek first, elevator algorithm), Files (File Naming, File structure, File types, File access, File attributes, File operations), Directories(Hierarchical directory system, Pathnames), File system Implementations(File System layout, Implementing files, Implementing directories), Security (Threats, Attackers, Controlling access to resources).

UNIT IV: Practical 30 Hours

Basic Linux:
Navigating the File System(home directory, pwd, cd, the three types of Files, absolute pathnames, relative pathnames, mkdir, rmdir, ls, listing by modification and access times), General-Purpose Utilities(cal, date, who, uname, echo, bc, clear), Handling Ordinary Files(cat, cp, rm, mv, more, the file command, wc, split, cmp, comm, diff, touch), Basic File Attributes(File permissions, using chmod to change relative and absolute permissions, chown), the Shell's Wild Cards, escaping, quoting, Redirection using <, >, >>, /dev/null, /dev/tty, Pipes using |, Command Substitution, shell variables, Simple Filters(pr, head, tail, cut, paste, sort, uniq, tr).

Advanced Linux*:*
Regular expressions(grep, Basic Regular Expressions, The Character Class, the *, the dot, Specifying pattern locations), Essential Shell Programming(Shell scripts, the bash command, read, using command line arguments, exit and exit status of a command, the logical operators && and ||, the if conditional, the case conditional, looping using while and for, positional parameters, test command for Numeric comparison, String comparison, File tests), Advanced Shell Programming(Computation using let, String handling using # and % in ${}).

List of practical assignments. (Problems may not be restricted to this list)

Basic practical shell scripts:

1. Write a shell script that will display all the multiples of 5 between 5 and 100.
2. Write a script that will accept a filename from the keyboard and determine whether the file exists. If the file exists, then its contents will be displayed else an error message will be displayed.
3. An integer is input through the keyboard. Write a script to find out whether it is an odd or even number.
4. Write a shell script that displays a list of all files in the current directory to which you have read, write and execute permissions.
5. Write a shell script that will display the multiplication table of any given number.
6. Write a script that converts a decimal number to a hexadecimal number. [Hint: use bc]
7. The length and breadth of a rectangle and radius of a circle are input through the keyboard. Write a script to calculate the area and perimeter of the rectangle as well as the area and circumference of the circle.
8. Write a shell script that will prompt the user to enter a character. The script will then determine whether the user entered a lowercase letter, an uppercase letter, a digit or a special symbol.
9. If the cost price and selling price of an item is input through the keyboard, write a script to determine whether the seller has made profit or incurred loss. Also determine how much profit was made or loss incurred.
10. Write a script that accepts a string inputted though the keyboard and echoes a suitable message if it does not have at least 10 characters.
11. The marks obtained by a student in five different subjects are input through the keyboard. The student gets a division as per the following rules:
    i. Percentage above or equal to 60 – First division
    ii. Percentage between 50 and 59 – Second division
    iii. Percentage between 40 and 49 – Third division
    iv. Percentage less than 40 – Fail
    Write a script to find the division obtained by the student.
12. Write a script that will accept two file names from the command line, copy the first file to the second file and then display the contents of the combined file. Proper error message should be displayed in case the copy is not successful.

13. Peter’s basic salary is input through the keyboard. His dearness allowance is 40% of his basic salary, and house rent allowance is 18% of the basic salary. Write a script to calculate his gross salary.
14. Write a script that accepts a filename as argument and displays the last modification time if the file exists, and a suitable message if it doesn’t.
15. Write a shell script, which receives any year from the keyboard and determines whether the year is a leap year or not. If no argument is supplied the current year should be assumed.
16. In a company, an employee is paid as follows: If his basic salary is less than Rs. 5000, then HRA = 10% of basic salary and DA = 90% of basic salary. If his salary is either equal or above Rs. 5000, then HRA = Rs. 900 and DA = 98% of basic salary. If the employee’s salary is input through the keyboard, write a script to find his gross salary.

Advanced practical shell scripts:

17. The distance between two cities (in km.) is input through the keyboard. Write a script to convert and print this distance in meters, feet, inches and centimetres.
18. Write a program that checks if any of a list of users given on the command line is logged in. For each user it should say whether he/she is logged in or not.
19. Write a script to calculate overtime pay of 10 employees. Overtime is paid at the rate of Rs. 12 per hour for every hour worked above 40 hours. Assume that employees do not work for fractional part of an hour.
20. Write a script that will receive any number of filenames as arguments. The script should check whether such files already exist. If they do, then it should be reported. If these files do not exist then:
21. Check if a sub-directory called mydir exists in the current directory. If it doesn’t exist then it should be created and in it, the files supplied as arguments should get created.
22. If mydir already exists then it should be reported along with the number of files that are currently present in mydir.
23. If a five digit number is input through the keyboard, write a script to calculate the sum of its digits.
24. Write a script that accepts two directory names as arguments and deletes those files in the second directory that are identical to the files in the first.
25. Write a shell script to find the factorial of any number entered through the keyboard.
26. Write a script that will read a filename from the command line and will change the name of the file to filename.aa1 where aa1 is the login_name of the user. (E.g. if the filename is Lucky and the user’s login_name is harry then, the filename will be changed to Lucky.harry).
27. Two numbers are entered through the keyboard. Write a script to find the value of one number raised to the power of another.
28. Write shell script to convert file names from UPPERCASE to lowercase file names or vice versa.
29. Write a script to print all prime numbers between 1 and 150.

30. Write a shell script that backs up all files in a directory into a Backup directory for every day of the week. In other words, on Monday all files go in a "Monday" or "1" backup directory, on Tuesday they all go into a "Tuesday" directory, and so forth. If a directory for today already exists, overwrite the files in it, otherwise create the directory.

Instructions to Paper Setter (Theory)

Questions should be set according to the following scheme.

Unit	**Questions to be set**	**Questions to be answered**
I	2	1
II	2	1
III	2	1
Total	**6**	**3**

Instructions to Paper Setter (Practical)

Questions should be set according to the following scheme.

Unit	**Questions to be set**	**Questions to be answered**	**Marks**
Section A (Basic)	2	1	8
Section B (Advanced)	2	1	11
Total	5	3	19

Exam Duration:

Theory	3 Hours
Practical	3 Hours

Suggested Readings

Text Books:

1. A. S. Tanenbaum and Herbet Bos, *Modern Operating Systems*, 4th Edition, Pearson, 2014.
2. S. Das, *UNIX Concepts and Applications,* 4th Edition, Tata McGraw-Hill, 2017.

Reference Books:

1. W. Stallings, *Operating Systems: Internals and Design Principles*, 9th Edition, Pearson, 2018.

2. A. Silberschatz, P. B. Galvin, and G. Gagne,*Operating System Concepts*, 8th Edition, New York: John Wiley and Sons, 2017.
3. Daniel J. Barrett, *Efficient Linux at the Command Line*, 1st Edition, O'Reilly, 2022.
4. Cameron Newham, *Learning the Bash Shell*, 3rd Edition, O'Reilly, 2005.

CSC-253: Computer System Architecture
(Contact Hours: 60, Credits: 4)

(Marks: 75 (External) + 25 (Internal)=100)

Course Objective

- CO1: The course aims to equip students with a solid foundation in computer system architecture, by providing a comprehensive understanding of the fundamental concepts, principles, and components of computer systems.
- CO2: Understanding Basic Computer Architecture, Memory Hierarchy, Input/Output Systems, System Bus and Interconnects, Pipelined architecture and Design of pipeline processor

Learning Outcomes

After completion of the course, the students are expected to be able to

- LO1: Explain the basics of organizational and architectural issues of a digital computer.
- LO2: Describe various data transfer techniques in digital computer and the I/O interfaces.
- LO3: Analyze the performance of various classes of Memories, build large memories using small memories for better performance and analyze arithmetic for ALU implementation.
- LO4: Describe the basics of hardwired and micro-programmed control of the CPU, and pipelined architectures.

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Basic concepts and Assembly language	15	18	25
II	Control Unit and Input output Organization	15	19	
III	Pipelining and vector processing	15	19	
IV	Memory Management	15	19	
Total		60	75	25

Detailed Syllabus

UNIT I :Basic concepts and Assembly language **15 Hours**

Functional units of a computer, Basic operational concepts, Bus structures, memory locations, Address and Encoding of Information, memory operations, Instructions and Instruction sequencing: Instruction Execution and straight line sequencing, Branching, Addressing modes, Assembly Language, Assembly and execution of programs.

UNIT II: Control Unit and Input-output Organization 15 Hours

Control memory, Address sequencing: Routine, mapping, conditioned Branching, mapping instruction, Design of control unit: Microprogram sequencer.

Input Output Organization: Peripheral Devices, Input-Output Interface, Modes of Transfer: Programmed I/O, Interrupt initiated I/O and Direct Memory Access (DMA). Memory mapped I/O vs. Isolated I/O.

UNIT III: Pipelining and vector processing 15 Hours

Four models of Computers (SISD, SIMD, MISD, MIMD), Parallel processing, pipeline. Arithmetic pipeline, Instruction pipeline, RISC pipeline, Vector processing; vector operations, Memory Interleaving.

UNIT IV: Memory Management 15 Hours

Memory Hierarchy: auxiliary memory, cache memory, multiprogramming; main memory, RAM, ROM, Bootstrap loader, computer start up. RAM and ROM chips, memory Address map, memory connection to CPU; Associative memory; cache memory; Locality of reference, hit ratio, mapping, Associative mapping, Direct mapping, writing into cache; virtual memory; Address space and memory space, Address mapping using pages, Associative memory page table; memory management hardware, memory protection

Instructions to Paper Setter

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
IV	2	1
Total	**8**	**4**

Exam Duration

Theory	3 Hours
Practical	-

Suggested Readings:

Texts:

1. Carl Hamacher, ZvonksVranesic, SafwatZaky, *Computer Architecture And Organization*, 5th Edition, McGraw Hill Educationi, 2011.
2. M. Morris Mano, *Computer System Architecture*, 3rd Edition, Pearson, 2017.

References:

1. H. Stone, *Introduction to Computer architecture*, 3rd Edition, Galgotia Publishing Ltd., 2001
2. P. Pal Chaudhuri, *Computer Organization and Design*, 4th Edition, Prentice Hall India, 2002.
3. Thomas C. Bartee, *Computer Architecture and Logic Design*, Tata McGraw-Hill, International Edition, 2001.
4. B. Ram, *Computer Fundamentals, Architecture and Organization*, 3rd Edition, New Age International Publishers, 2002.
5. S. Salivahanan, S. Arivazhaga, *Circuits And Design*, 5th Edition, Oxford Higher Education, 2018.

CSC-300: Software Engineering
(Contact Hours: 60, Credits: 4)

(Marks: 75 (External) + 25 (Internal)=100)

Course Objectives

- CO1: To provide the idea of decomposing the given problem into Analysis, Design, Implementation, Testing and Maintenance phases.
- CO2: To provide an idea of using various process models in the software industry according to given circumstances.
- CO3: To gain knowledge of how Analysis, Design, Implementation, Testing and Maintenance processes are conducted in a software project.

Learning Outcomes

- LO1: Determine the primary problems that impact all software development processes.
- LO2: Choose relevant software development process models, methodologies, and strategies for managing a specific software development process, and justify the choices
- LO3: Implement different software estimation metrics such as cost, effort size, staffing etc.
- LO4: Describe various software design approaches and various coding and testing strategies used in software engineering principles
- LO5: Know about software reliability and how to calculate software maintenance cost

Outline of the Course

Unit	**Topic**	**Minimum Class Hours**	**ExternalMarks**	**Internal Marks**
I	Introduction To Software Engineering Software life cycle models	14	20	25
II	Software project management; requirements analysis and specification; software design	18	20	
III	Function Oriented Software Design, Object Modelling using UML, Object Oriented Software Development and user interface Design	18	20	
IV	Testing, software reliability and maintenance	10	15	
Total		**60**	**75**	**25**

Detailed Syllabus

UNIT I: Introduction to Software Engineering Software Life Cycle Models 14 Hours

Introduction: Evolution of an art to an engineering disciple, Solution to the software crisis, Computer systems engineering, What is Software Engineering? Why Study Software Engineering, What is Software? Characteristics of Software, The evolving role of software, Changing Nature of Software, legacy software, Software myths

Software Life cycle models: Importance of a life cycle model, waterfall model (feasibility study, requirement analysis and specification, design, coding and unit testing integration and system testing, maintenance), prototyping model, evolutionary model, spiral model, Agile Model, Agile V/s traditional SDLC Models, SCRUM model, Advantages and disadvantages of each of these SDLC models

UNIT II: Software project management; Requirements Analysis and Specification; Software Design 18 hours

Software project management: Responsibilities of a software project manager, project planning, project estimation techniques, COCOMO, Scheduling (work breakdown, Activity Networks and critical Path Method, Gantt Charts, PERT Charts, project Monitoring and control), Organization and team structures (Organization structure, team structure), Risk management (Risk identification, risk assessment, Risk Containment), Software configuration Management (Necessity of software Configuration Management, Configuration Management Activities, Source code control system and RCS.

Requirement Analysis and specification: Requirement gathering and analysis, Software requirements specification (Content of the SRS document, Functional Requirements, how to identify the functional requirements, how to document the functional requirements, traceabilty, characteristics of a good SRS document, techniques for representing complex logic).
Software Design: introduction of a good software design, cohesion and coupling (classification of cohesiveness and coupling), Software designs approaches (function-oriented design, Object-oriented Design).

UNIT III: Function Oriented Software Design, Object Modelling using UML, OO Software Development and User Interface Design 18 Hours

Function Oriented Software Design: Overview of SA/SD methodology, Structured Analysis, data Flow Diagrams (DFDs)(primitive symbols used for constructing DFDs, important concepts associated with designing DFDs, developing the DFD Model of a system, Shortcomings of the DFD Model), Extending DFD technique to real-time systems, Structured Design (flow chart vs. structure chart, transformation of a DFD model into a structure chart), Detailed Design, Design Review.

Object Modelling using UML: Overview of Object-Oriented Concepts, Unified Modeling Language (UML), UML diagrams, USE CASE Model, Class Diagrams, Interaction Diagrams, Activity diagrams.
User Interface Design: characteristics of a good user interface, basic concepts (user guidance and online help, mode-based vs. Modeless Interface, Graphical User Interface (GUI) vs. Text-based User Interface), Types of user interfaces (command language-based Interface, Menu-based

Interface, direct manipulation Interface), Component-Based GUI Development (Window system, Types of widgets, Visual programming), User interface methodology (Design, a GUI design methodology, Task and object modeling, selecting a metaphor, interaction design and rough layout, user interface inspection).

UNIT IV: Testing, Software Reliability and Maintenance 10 Hours

*Coding and Testing:*coding standards and guidelines, code review (code walkthroughs, code inspection, clean room testing, software documentation), Testing (testing, verification vs. validation, design of test cases), Testing in the large, Testing in the small, unit testing, black-box testing, White-box testing, debugging, program analysis tools, integration testing, system testing.
Software Reliability: Software reliability, statistical testing, software quality, and software quality management.

Software Maintenance: Characteristics of software maintenance (types of software maintenance, characteristics of software evolution, special problems associated with software maintenance), software reverse engineering, software maintenance process models, estimation of maintenance cost.

*Software Reuse:*What can be Reused?, Why Almost no Reuse so far?, Basic Issues in Any Reuse Program, A Reuse Approach

Instructions to Paper Setter

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
IV	2	1
Total	**8**	**4**

Exam Duration

Theory	3 hours
Practical	-

Suggested Readings

Text:

1. Rajib Mall, *Fundamentals of Software Engineering*, 5th Edition, Pearson Education/Prentice Hall of India, New Delhi, 2018
2. Roger S Pressman, *Software Engineering A Practitioner's Approach*, 9th Edition, McGraw Hill International Edition, 2019

References:

3. Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli, *Fundamentals Of Software Engineering*, 2nd Edition, Prentice Hall of India Private Limited, New Delhi, 2002.
4. Richard E Fairley, *Software Engineering Concepts*, Indian Edition,Tata McGraw Hill Publishing Company Limited, New Delhi, 2017.

CSC-301: Mathematical Foundations of Computer Science
(Contact Hours: 60, Credits: 4)

(Marks: 75 (External) + 25 (Internal)=100)

Course Objectives

- CO1: Develop a solid understanding of foundational mathematical concepts relevant to computer science, including linear algebra, discrete mathematics, and probability theory.
- CO2: These objectives aim to provide students with a strong foundation in mathematics and statistics necessary for tackling advanced topics in computer science and for solving real-world problems in data analysis, machine learning, and algorithm design.

Learning Outcome

- LO1: Recall and explain fundamental mathematical concepts such as linear algebra, discrete mathematics, and probability theory.
- LO2: Demonstrate understanding of the relationships between mathematical concepts and their applications in computer science, such as the use of matrices in computer graphics or the role of probability in machine learning algorithms.
- LO3: Apply mathematical techniques to solve computational problems in computer science, such as optimization problems, algorithm analysis, and graph algorithms
- LO4: Analyze the efficiency and complexity of algorithms using mathematical tools such as asymptotic notation, recurrence relations, and graph theory

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Sets, Functions and Relations	10	13	25
II	Combinatorics and Recurrence relations	10	12	
III	Basic Linear Algebra and Graph Theory	20	25	
IV	Basic Statistics and Probability	20	25	
Total		**60**	**75**	**25**

Detailed Syllabus

Unit I: Sets, Functions and Relations **10 Hours**

Sets – Definitions, Representations, Power Set, Cartesian Product, Set operations, Set Identities

Functions – Definitions and terminologies, One-to-One and Onto, Inverse Functions and Compositions of Functions
Relations - Relations on One Set, Relations and Directed Graphs, Properties of Relations, Equivalence Relations, Partitions, Partial Orders, Directed Acyclic Graphs, Partial Orders and Total Orders

Unit II: Combinatorics and Recurrence Relations **10 Hours**

Combinatorics: Basics of counting, Combinations & Permutations, Binomial Coefficients, principles of Inclusion-Exclusion.

Recurrence relations: Generating Functions, Function of Sequences, Calculating Coefficients of generating functions, solutions of recurrence relations by substitution and generating functions, solution of non-recurrence relations by conversion to linear recurrence relations.

Unit III: Basic Linear Algebra and Graph Theory **20 Hours**

Linear Algebra: System of equations, matrices and vectors, Singular vs non-singular matrices, Linear dependence and independence, Determinant of a matrix, Operations on matrices and vectors – Addition and Difference of matrices, Dot product, multiplication of a matrix by a vector, Matrix multiplication; Identity matrix, Inverse of a matrix, linear transformation of matrices, Singularity and rank of linear transformations, Bases and Span, Eigenvalues and Eigenvectors

Graph Theory – Graphs, Graph variations, applications of graphs, isomorphism, connectivity, distance, diameter, degree, walks, adjacency matrix, trees, spanning trees; Graph colouring, bipartite graphs, planar graphs

Unit IV: Basic Statistics and Probability **20 Hours**

Basic Statistics: Frequency Distributions, Graphs of Frequency Distributions, Descriptive Measures, Quartiles and Percentiles

Probability: Sample Spaces and Events, Counting, Probability, The Axioms of Probability, Some Elementary Theorems, Conditional Probability, Bayes' Theorem

Probability Distributions: Random Variables, The Binomial Distribution, The Mean and the Variance of a Probability Distribution, The Poisson Distribution, The Geometric distribution, Simulation of random variables-Monte Carlo method

Probability Densities: Continuous Random Variables, The Normal Distribution, Joint Distributions - Discrete and Continuous

Sampling Theory: Introduction – Population and samples – Sampling distribution of Means and Variance (definition only) – Central limit theorem (without proof)

Instructions to Paper Setter

Questions should be set according to the following scheme.

Unit	**Questions to be set**	**Questions to be answered**
I	2	1
II	2	1
III	2	1
IV	2	1
Total	**8**	**4**

Exam Duration

Theory	3
Practical	-

Suggested Readings

Texts:

1. Kenneth Rosen, *Discrete Mathematics and Its Applications*, 8th Edition, McGraw Hill, 2021
2. Richard A. Johnson, Irwin Miller, John Freund, *Probability and Statistics for Engineers*, 8th Edition, Pearson, 2015
3. David C. Lay, Steven R. Lay, Judi J. McDonald, *Linear Algebra and its Applications*, 5th Edition, Pearson Education, 2023

References:

1. Trembly, J. P., P. Manohar, Discrete Mathematical Structures With Applications to Computer Science, 1st Edition, Tata McGraw-Hill, 2017
2. Sheldon Ross, A First Course in Probability, 10th Edition, Pearson Education, 2022
3. C.L. Liu , D.P. Mahapatra, Elements of Discrete Mathematics, 4th Edition, McGraw Hill, 2017,
4. J. L. Hein, Discrete Structures, Logic, and Computability, 3rd Edition, Jones and Bartlett Publishers, 2010
5. D.J. Hunter, Essentials of Discrete Mathematics, 1st Edition, Jones and Bartlett Publishers, 2016

CSC-302: Programming in Python
(Contact Hours: 75, Credits: 3+1 = 4)

(Marks: 56(T)+19(P)=75 (External) + 25 (Internal)=100)

Course Objectives

- CO1: To introduce students to the fundamentals of Python programming language.
- CO2: To develop critical thinking and data analysis skills using Python.
- CO3: To introduce students to object oriented programming principles and apply them for problem solving.
- CO4: To equip students with the ability to apply Python programming in various domains..
- CO5: To prepare students for further study or professional work involving Python.

Learning Outcomes

By the end of this course, students should be able to:
- LO1: Understand the basic syntax and semantics of Python programming language.
- LO2: Develop a deeper understanding of programming concepts such as loops, conditionals, functions, and data structures.
- LO3: Write Python programs to solve simple to moderately complex problems.
- LO4: Design programs with object-oriented principles.
- LO5: Apply Python libraries and modules for specific tasks.

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Basics of Python Programming	15	18	19
II	NumPy, Strings and Data Structures	15	19	
III	Exception Handling, File Handling, Object Oriented Programming	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

Detailed Syllabus

Unit I: Basics of Python Programming **15 Hours**

Introduction to Python: Overview of Python programming language, features, and applications in domains such as web development, data science, and machine learning.
Basic syntax: Python character set, Python tokens (keyword, identifier, literal, operator, punctuator), variables, constants, DataTypes (Numbers (integers, floating point, boolean), Sequence(string, list, tuple), None, Mapping(Dictionary)), mutable and immutable data types, Operators (arithmetic, comparison, assignment, unary, bitwise, shift, logical, membership, identity), operator precedence and associativity. Using functions input() and print().

Control Statements: if-elif-else for decision-making; iteration using for and while, range(); break, continue and pass statements.

Functions and modules: Need for functions, Defining functions with parameters and returning values, required arguments, keyword arguments, default arguments, variable length arguments, lambda functions. Scope and Lifetime: local and global variables, global statement, resolution of names, flow of execution. Modules: Definition of module, advantages of modules, from … import statement, user-defined modules, modules and namespaces, function redefinition.

Unit II: NumPy, Strings and Data Structures 15 Hours

NumPy: NumPy as a Python array implementation, creating arrays from existing data, array attributes, filing arrays with values, reading arrays from ranges, array operators, NumPy calculation methods, indexing and slicing, shallow and deep copying, reshaping and transposing arrays.

Strings: String concatenating, appending, multiplying, slice operation; string formatting operator, string methods (capitalize(), count(), startswith(), endswith(), find(), isalpha(), isdigit(), islower(), isupper(), isspace(), len(), lower(),upper(), strip(), replace(), split(), join()), in and not in operators, comparing strings using operators.

Lists: Creating a list, adding and updating values, cloning and nesting lists, list operations (len, concatenation, repetition, in, not in, min, max, sum, all, any, sorted), list methods (append(), count(), insert(), pop(), remove(), reverse(), sort(), extend()), enumerate()), indexing and slicing.

Tuples: Creating a tuple, adding, updating and deleting values, tuple operations (len, concatenation, repetition, in, not in, min, max), tuples for returning multiple values, nested tuples, variable length argument tuples, indexing and slicing.

*Sets***:** Creating a set, operations (update(), add(), remove(), pop(), clear(), len(), in, not in, issubset(), issuperset(), union(), intersection(), difference(), copy(), enumerate(), min(), max(), sum(), sorted()).
Dictionary: Creating a dictionary, adding, updating and deleting elements, dictionary methods (len(), clear(), copy(), fromkeys(), get(), items(), keys(), update(), values(), in, not in).

Unit III: Exception Handling, File Handling, Object-Oriented Programming 15 Hours

Exception handling: Errors and exceptions, Handling exceptions using try-except blocks, multiple except blocks, multiple exceptions in a single block, else clause with except, raising exceptions with raise statement, re-raising an exception, finally block.

File handling: File types: Text and binary files, opening and closing files, text file access modes (read, write, append) writing/appending data to a text file using write() and writelines(), reading from a text file using read(), readline() and readlines(), renaming and deleting files, file position using seek(), file methods (fileno(), flush(), isatty(), truncate(), rstrip()), directory methods (mkdir(), getcwd(), chdir(), rmdir()).

Object Oriented Programming: OOP principles, defining classes and objects in Python, Creating and initializing objects, Instance variables and methods, Class variables and methods, Getter and setter methods. Access specifiers: public, private, and protected, Types of Inheritance, Superclass and subclass relationships, Method overriding and inheritance hierarchy, Polymorphism and method overriding, garbage collection.

Unit IV:Practical **30 Hours**

List of practical assignments. (Problems may not be restricted to this list)

1. Write a program that will accept a number from the user and check if the number is even or odd.
2. Write a program that will find the greatest amongst three numbers.
3. Write a program to check if a number is an Armstrong number.
4. Write a program to find simple interest using a user defined function with parameters and with return value.
5. Write a program to accept a word and display the number of letters using functions.

6. Write a program to find the sum of all numbers in a list using functions.
7. Write a program to check if a given number is a palindrome number or not.
8. Write a program to input names and salary of 10 people and store them in a file.
9. Write a program to append the string "Hello this file has been appended" to an existing file.
10. Write a program to store the contents of a list into a file.
11. Write a program to read an existing file and display it, the replace all the letters "a" with the letter "z" and display the file again with the new changes.
12. Write a function to find the sum of all elements in a NumPy array.
13. Write a function to find the sum, difference and product of two matrices using NumPy.
14. Write a program to sort a NumPy array in ascending order.
15. Write a program to compute mean, median, variance and standard deviation for a NumPy array.
16. Write a program to extract a sub-array from a given NumPy array using slicing and indexing.
17. Write a program that takes a list of numbers as input and returns a new list containing only the even numbers from the original list.
18. Write a program that prompts the user to enter two lists and then concatenates them into a single list. Ensure no duplicates are present in the final list.
19. Write a program that sorts a list of strings based on the length of each string, from shortest to longest.
20. Write a program that takes two sets as input and returns a new set containing only the common elements between the two sets.
21. Write a program that converts a list to a set and removes duplicates from the list.
22. Create a program that prompts the user to enter two sets and then performs union, intersection, and difference operations on them.

23. Write a program that takes a tuple and returns a new tuple containing only its first half elements.
24. Write a program that takes a dictionary of student names and their corresponding scores, and returns the student with the highest score.
25. Write a program that prompts the user to enter two dictionaries and merges them into a single dictionary. If there are common keys, combine their values.
26. Write a program that removes all entries from a dictionary with a value less than a specified value.
27. Write a program that demonstrates shallow copy using lists. Create a list containing nested lists and then create a shallow copy of it. Modify one of the nested lists in the original list and observe the changes in the shallow copy.
28. Write a program that showcases deep copy using dictionaries. Create a dictionary containing nested dictionaries and then create a deep copy of it. Modify one of the nested dictionaries in the original dictionary and observe if the changes reflect in the deep copy.
29. Design a class representing a bank account. Include attributes such as account number, account holder name, balance, and methods for deposit, withdrawal, and checking the balance.
30. Create a base class called Shape with methods to calculate area and perimeter. Implement derived classes Rectangle and Circle inheriting from Shape. Override the area and perimeter methods in each derived class.
31. Design a class representing a calculator with methods for addition, subtraction, multiplication, and division. Implement error handling to handle divide by zero and invalid input cases.

Instructions to Paper Setter (Theory)

Questions should be set according to the following scheme:

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
Total	**6**	**3**

Instructions to Paper Setter (Practical)

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered	Marks
I	2	1	7
II	3	2	6+6=12
III			
Total	**5**	**3**	**19**

Exam Duration

Theory	3 Hours
Practical	3 Hours

Suggested Readings

Texts:

1. Reema Thareja, *Python Programming Using Problem Solving Approach*, 1st Edition, Oxford University Press, India, 2017.
2. Martin C. Brown, *Python: The Complete Reference*, 4th Edition, McGrawHill, 2018.

References:

1. Paul J. Deitel, Harvey M. Deitel, *Intro to Python for Computer Science and Data Science*, 1st Edition, Pearson, India, 2022.

CSC-350: Design and Analysis of Algorithms
(Contact hours: 60. Credits: 4)

(Marks: 75 (External) + 25 (Internal)=100)

Course Objectives

- CO1: The objective of this course is to study paradigms and approaches used to analyse and design algorithms.
- CO2: It also ensures that students understand how the worst-case time complexity of an algorithm is defined, and how asymptotic notation is used to provide a rough classification of algorithms.
- CO3: Another objective is to learn how to compare algorithms and to know that there are still many problems for which it is unknown whether there exist any efficient algorithms.

Learning Outcomes

- LO1: Students will learn pseudo code for expressing algorithms, performance analysis, space complexity, time complexity, asymptotic notation, big oh notation, omega notation, theta notation, and little oh notation, probabilistic analysis, amortized analysis.
- LO2: Students will learn about different algorithm design techniques.
- LO3: Students will also learn about NP complete and NP hard problems.

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Review of elementary data structures, Introduction to analysis of algorithms	10	15	25
II	Algorithms Design Techniques	20	25	
III	Graph based Algorithms, Dynamic algorithms	20	25	
IV	Intractable Problems	10	10	
Total		**60**	**75**	**25**

Detailed Syllabus

UNITI: Review of elementary data structures, Introduction to analysis of algorithms 10 Hours

Review of elementary data structures (stack, linked list, queue, tree, graph), basics of computational models- Random access model, Turing machine. Introduction to analysis of algorithms, asymptotic notations, analysis of recursive programs, solving recurrence relations.

UNITII: Algorithms Design Techniques **20 Hours**

Introduction to Algorithms Design Techniques,Divide and Conquer approach: structure of divide-and-conquer algorithms, quick sort, merge sort, binary search, Strassen's matrix multiplication problem, sets and disjoint sets, union and find algorithms. Greedy Approach: activity selection problem. Minimum Spanning trees: Prim's algorithm & Kruskal's algorithm, Huffman codes, Hill climbing algorithm, Knapsack problem. Backtracking: 8-Queen Problem, graph coloring, Hamiltonian cycles, Introduction to Branch and bound.

UNITIII: Graph-based Algorithms, Dynamic Algorithms **20 Hours**

*Graph based algorithm:*Depth-first search, Breadth-first search, strongly connected components; shortest path problems, Dijkstra's algorithm, Floyd-Warshall's algorithm, Bellmen ford algorithm. *Dynamic programming:* Overview, difference between dynamic programming, divide and conquer and greedy approach. Longest Common sequence, Matrix chain multiplication.

UNITIV: Intractable Problems **10 Hours**

Intractable Problems and basic Concepts, Nondeterministic Algorithms, NP Completeness,Cook's Theorem, Examples of NP-Hard and NP-Complete problems. Problem Reduction.

Instructions to Paper Setter

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
IV	2	1
Total	**8**	**4**

Exam Duration

Theory	3
Practical	-

Suggested Readings

Texts:

1. T. H.Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein,*Introduction to Algorithms*, 3rdEdition, MIT Press, 2010.
2. A. V. Aho, J. E. Hopcroft, and J. D. Ullman,*The Design and Analysis of Computer Algorithms*, 1st Edition, Pearson, 2002.

References

1. K.S.Basu,*Design Methods and Analysis of Algorithms*, 2ndEdition, Prentice-HallIndia, 2015.
2. E.Harwitz, and S. Sahani,*Fundamentals of Computer Algorithms*, 2nd Edition, ComputerScience Press,2009.

CSC-351: Artificial Intelligence
(Contact Hours: 60, Credits: 4)

(Marks: 75 (External) + 25 (Internal)=100)

Course Objectives

- CO1: Develop a comprehensive understanding of the fundamental principles and techniques of artificial intelligence, including search strategies, knowledge representation, expert systems and natural language processing
- CO2: Students will gain insights into the theoretical foundations and practical applications of AI, enabling them to analyze and solve complex problems using AI techniques.

Learning Outcome

- LO1: Identify and recall various applications of artificial intelligence across different domains and the basic principles and strategies of search algorithms in AI.
- LO2: Explain the functioning of different search strategies and their applications in problem-solving.
- LO3: Demonstrate understanding of game playing strategies and their relevance in developing AI agents.
- LO4: Analyze the advantages and limitations of different knowledge representation techniques in AI.
- LO5: Evaluate the effectiveness of expert systems developed using various knowledge representation techniques in solving domain-specific problems.

Outline of theCourse

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Overview of AI and Basic Search Strategies	15	18	25
II	Search Strategies for AI Production Systems	15	19	
III	Knowledge Representation	15	19	
IV	Introduction to Neural Networks and Expert Systems	15	19	
Total		**60**	**75**	**25**

Detailed Syllabus

Unit I: Overview of AI and Basic Search Strategies **15 Hours**

*Introduction to AI:*The AI Problems, The Underlying Assumption, AI Techniques, The Level of the Model, Criteria for Success, AI Applications.
Problem *solving, Search and Control Strategies:*Defining the Problem as a State Space Search, Production Systems, Control Strategies, Breadth-First Search, Depth-First Search.

Unit II: Search Strategies for AI Production Systems 15 Hours

Heuristic Search Techniques: Generate-and-Test, Hill Climbing, Simple Hill Climbing, Steepest-Ascent Hill Climbing, Simulated Annealing, Best-First Search, OR-Graphs, the A* Algorithm, Problem Reduction, AND-OR Graphs, The AO* Algorithm, Constraint Satisfaction, Means-End Analysis.

Game Playing*:* Overview, The Minimax Search Procedure, Adding Alpha-Beta Cutoffs

Unit III: Knowledge Representation 15 Hours

Knowledge Representation Issues: Representations and Mappings, Representing Simple Facts in Logic, Knowledge Representation Attributes, Computable Functions and Predicates, Resolution, Conversion to Clause Form, the Basics of Resolution, Resolution in Propositional Logic, Procedural vs. Declarative Knowledge, Logic Programming, Forward vs. Backward Reasoning, Matching, Control Knowledge.

Statistical Reasoning*:* Probability and Bayes’ theorem, Certainty factors and Rule-Based Systems, Bayesian Networks, Dempster-Shafer Theory

Unit IV: Introduction to Neural Networks and Expert Systems 15 Hours
Learning in Neural Networks: Introduction to Neural Networks, Perceptron, Backpropogation (Introduction only), Applications of Neural Networks;

Expert System: Representing and Using Domain Knowledge, Expert System Shells, Explanation, Knowledge Acquisition.

Instructions to Paper Setter

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
IV	2	1
Total	**8**	**4**

Exam Duration

Theory	3 Hours
Practical	-

Suggested Readings

Texts:

1. E. Rich, K. Knight, S. B. Nair, *Artificial Intelligence*, 3rd Edition, Tata McGraw-Hill, 2017

References:

1. D. W. Patterson, *Introduction to Artificial Intelligence and Expert Systems*, 1st Edition, Pearson Education India, 2015
2. P. H. Winston, *Artificial Intelligence*, 1st Edition, Pearson Education Asia, 2002
3. S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th Edition, Pearson Education, 2022

CSC-352: Cryptography and Network Security
(Contact Hours: 60, Credits: 4)

(Marks: 75 (External) + 25 (Internal)=100)

Course Objectives:

- CO1: Understand fundamental concepts of information security.
- CO2: Understand the various encryption techniques and exploring different cryptographic algorithms.
- CO3: Understand security mechanisms such as firewalls and intrusions and security policies and standards.

Learning Outcomes:

- LO1: Identify and classify particular examples of attacks and factors driving the need for network security.
- LO2: Compare and contrast symmetric and asymmetric encryption systems
- LO3: Usage of network security tools and applications to understand the system level security
- LO4: Evaluate security mechanisms using rigorous approaches by key ciphers and Hash functions

Outline of the course:

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Information Security and Cryptography	15	15	25
II	Block Ciphers and Symmetric and Asymmetric key Cryptography	15	20	
III	Key Management and Message Authentication	15	20	
IV	Network Security and Web Security	15	20	
Total		**60**	**75**	**25**

Detailed Syllabus

Unit I: Information Security and basic concept of Cryptography 15 Hours

Information security: Attributes of Information Security: Confidentiality, Integrity, Availability. Security Attacks, Security Services and Security Mechanisms.

Cryptography: Secret key and public key cryptography, Classical Encryption Techniques: Symmetric cipher model - Cryptography and Crypanalysis; Substitution techniques- Caesar Cipher, Monoalphabetic Cipher, Playfair Cipher; Transposition technique – Rail fence, Simple and double columnar transposition.

Unit II: Block Ciphers and Symmetric and Asymmetric key Cryptography 15 Hours

Block Cipher Principles: Block and Stream Ciphers, The Feistel Cipher structure; DES, Modes of Block Cipher.

Public Key Cryptography: Principles of Public Key Cryptosystems, Applications, Requirements, Cryptanalysis, RSA Algorithm

Unit III: Key Management and Message Authentication 15 Hours

Key Management and Distribution: Symmetric Key Distribution, Diffie- Hellman Key Exchange, public key distribution; Authentication Requirements, authentication Function, Message Authentication Code(MAC), Hash function; Digital Signatures – Requirements, Direct Digital Signature, Arbitrated Digital Signature; Digital signature Standard.

Unit IV: Network Security and Web Security 15 Hours

Network Security Applications: Kerberos–Version 4; IP Security Architecture– IPSec Documents, IPSec Services, Security associations, Transport and Tunnel Modes;

Web Security: SSL, Intruders and Viruses, Firewall Design Principles – Firewall Characteristics, Types of Firewalls: Packet Filtering Router.

Instructions to Paper Setter

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
IV	2	1
Total	**8**	**4**

Exam Duration

Theory	3 hours
Practical	-

Suggested Readings

Texts:

1. W. Stallings, *Cryptography & Network Security Principles & Practices*,8th Edition, Pearson Education, 2023.
2. D. R. Stinson, *Cryptography: Theory & Practice*, 4thedition, CRC Press, 2017.

References:

1. W. Stallings, *Networks Security Essentials: Application & Standards*, 6th Edition, Pearson Education, 2018.
2. B. L. Menezes, *Network Security & Cryptography*, 1st Edition, Cenegage Learning, 2011.
3. Brijendra Singh, *Network Security and Management*, 1st Edition, PHI, 2012.

CSC-353: Machine Learning
(Contact Hours: 75, Credits: 3+1=4)

(Marks: 56(T)+19(P)=75 (External) + 25 (Internal)=100)

Course Objectives

- CO1: Understand core concepts like supervised vs unsupervised learning, model evaluation (accuracy, overfitting, underfitting), data preparation (cleaning, pre-processing), and feature engineering (construction, extraction, selection).
- CO2: Explore various classification algorithms (k-Nearest Neighbors, Decision Trees, Random Forests, Naive Bayes, Logistic Regression) and regression models (Simple Linear, Multiple Linear). Apply these techniques to analyze relationships within data.
- CO3: Grasp the principles of unsupervised learning methods like k-Means Clustering, Hierarchical Clustering, and Principal Component Analysis (PCA) for data exploration and dimensionality reduction.
- Co4: Gain a foundational understanding of the biological inspiration behind neural networks, their basic components (activation functions, weights, biases), the training process (loss functions, backpropagation, optimization), and common challenges like overfitting and bias.

Learning Outcomes

- LO1: Define key machine learning terms like supervised learning, unsupervised learning, and reinforcement learning; Explain the concept of model bias and variance; Compare and contrast different machine learning algorithms; Describe the applications of machine learning in various fields.
- LO2: Pre-process a given dataset; Implement a machine learning model from scratch;
- LO3: Interpret the results of a machine learning model and identify potential issues;
- LO4: Communicate the findings of a machine learning project, including the chosen model, its performance, and limitations.

Outline of the course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Foundations of Machine Learning	15	18	19
II	Feature Engineering and Supervised Learning (Classification)	15	18	
III	Supervised Learning (Regression), Unsupervised Learning, and Conceptual Overview of Neural Networks	15	20	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

Detailed Syllabus

Unit I: Foundations of Machine Learning **15 Hours**

Introduction to Machine Learning: Human Learning; Machine Learning (Well-posed learning problem, Types, Applications, and Issues).

Data Preparation for Modelling: Basic Data types in Machine Learning; Exploring -numerical data, categorical data, and relationship between variables; Data issues and remediation; Data pre-processing.

Modelling and Evaluation: Selecting a model; Training a model- Holdout method, K-Fold Cross Validation, Bootstrap Sampling; Model representation and interpretability- under-fitting, over-fitting, bias - variance trade-off; Model performance evaluation for- classification, regression, and clustering.

Unit II: Feature Engineering and Supervised Learning (Classification) **15 Hours**

Feature Engineering: Feature *construction*, Feature extraction, Feature Selection

Machine Learning with Probabilistic Reasoning:*Review*of Bayes' Theorem (Prior and Posterior probability, likelihood); Bayesian Learning Methods -Naïve Bayes classifier.

Supervised Learning- Classification: Basics of supervised learning- classification, k-Nearest neighbor, decision tree, random forest.

Unit III: Supervised Learning (Regression), Unsupervised Learning, and Conceptual Overview of Neural Networks **15 Hours**

Supervised Learning- Regression: Simple linear regression, Multiple Linear Regression, Logistic Regression.

Unsupervised Learning: Basics of unsupervised learning; K-Means Clustering, Hierarchical clustering, Principal Component Analysis (PCA).

*Conceptual Overview of Neural Networks:*Biological inspiration, activation functions(ReLU, sigmoid-explained conceptually), parameters of the network (weights and biases), loss functions (conceptual overview), Backpropagation Algorithm (conceptual overview - not mathematical derivation), Optimization Techniques (concept of gradient descent), Overfitting and Underfitting - Challenges and Solutions (conceptually), bias in neural networks (basic understanding).

Unit IV: Practical **Hours: 30**

Language: Python

Datasets: Explore publicly available online repositories, such as UCI Machine Learning Repository (https://archive.ics.uci.edu/), Kaggle Datasets (https://www.kaggle.com/datasets), OpenML (https://www.openml.org/), etc but not restricted to just these sources.

To evaluate the performance of the regression/classification models, you may run these experiments (as and when needed):

- Scale/Normalize the data
- Reduce dimension of the data with different feature selection techniques
- Split datasets into training and test sets and evaluate the decision models
- Perform k-cross-validation on datasets for evaluation
- Present the performance results of the machine learning models

List of practical assignments. (Problems may not be restricted to this list)

1. **Iris Flower Classification:**
   - **Dataset:** Iris Flower Dataset (https://archive.ics.uci.edu/dataset/53/iris)
   - **Question:** Can a decision tree be used to classify different species of Iris flowers (Iris setosa, Iris versicolor, Iris virginica) based on features like sepal length, sepal width, petal length, and petal width?
   - **Question:** Can a Random Forest be used to improve the classification accuracy of Iris flowers (Iris setosa, Iris versicolor, Iris virginica) compared to a single decision tree, using features like sepal length, sepal width, petal length, and petal width?

2. **Wine Quality Prediction:**
   - **Dataset:** Wine Quality Dataset (https://archive.ics.uci.edu/dataset/109/wine)
   - **Question:** Can a decision tree be used to predict the quality (good or bad) of wine based on features like alcohol content, residual sugar, and free sulfur dioxide?

3. **Character Recognition (Kaggle MNIST Handwritten Digits):**
   - **Dataset:** https://www.kaggle.com/datasets/hojjatk/mnist-dataset
   - **Question:** Can a Random Forest be used to classify handwritten digits (0-9) based on pixel intensities in a 28x28 image? How does the accuracy compare to other algorithms (like K-Nearest Neighbors)?
   - **Question:** Can we use PCA to reduce the dimensionality of handwritten digit images (28x28 pixels) while preserving most of the information relevant for classification?

4. **Movie Recommendation System (UCI Machine Learning Repository - MovieLens):**
   - **Dataset:** https://archive.ics.uci.edu/datasets (Choose a version suitable for beginners)
   - **Question:** Can a Random Forest be used to recommend movies to users based on their past ratings (genres, actors, etc.)? How does the recommendation accuracy change with different hyperparameter settings (number of trees, maximum depth)?
   - **Question**: Can we use simple linear regression to predict movie ratings based on a single feature, like the year a movie was released? How well does this model perform compared to using the average rating?

5. **Predicting House Prices**
   - **Dataset:** https://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set
   - **Question:** Can we use multiple linear regression to predict house prices based on features like the average number of rooms per dwelling, crime rate, and distance to

highways? How does the model's performance compare to using just one feature (e.g., number of rooms)?

6. **Analyzing Student Grades (Using Kaggle Student Performance Dataset):**
   - **Dataset:** Choose a student performance dataset on Kaggle suitable for beginners (ensure it has multiple features besides grades).
   - **Question:** Can we use multiple linear regression to understand how factors like study hours, parental education level, and extracurricular activities influence student grades? How important is each feature according to the model (consider feature coefficients)?

7. **Spam Email Detection (Using UCI Machine Learning Repository - Spambase):**
   - **Dataset:** https://archive.ics.uci.edu/ml/datasets/Spambase
   - **Question:** Can we use logistic regression to classify emails as spam or not spam based on features like word frequency and presence of certain symbols? How well does the model perform based on metrics like accuracy and precision/recall?

8. **Movie Recommendation System Exploration (Using MovieLens Dataset - Choose a beginner-friendly version):**
   - **Dataset:** You can find various versions of the MovieLens dataset online (ensure it has features like user ratings and genres).
   - **Question:** Can we use k-means clustering to group movies based on their genres (action, comedy, etc.)? How can these movie clusters be used to recommend movies to users based on their past ratings (consider exploring further with collaborative filtering techniques)?

9. **Document Clustering:**
   - **Dataset:** Using UCI Machine Learning Repository – Twenty Newsgroup
   - **Question:** Can we use hierarchical clustering to group news articles from different topics (sports, politics, etc.) based on their word content? How does the resulting dendrogram visualize the hierarchical relationships between the articles?

Instructions to Paper Setter (Theory)

Questions should be set according to the following scheme:

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
Total	**6**	**3**

Instructions to Paper Setter (Practical)

Questions should be set according to the following scheme:

Unit		Questions to be set	Questions to be answered

IV (Practical)		5	3
		At least one question from Classification, one question from Regression, and one question from Unsupervised Learning	

Exam Duration

Theory	3 hours
Practical	3 hours

Suggested Readings

Texts:

1. S. Dutt, S. Chandramouli, and A. K. Das, *Machine Learning*, 1st Edition, Pearson India Education Services Pvt. Ltd, 2019.

References:

1. E. Alpaydin, *Introduction to Machine Learning*, 4th edition, The MIT Press, 2020.
2. S. Raschka, V. Mirjalili, *Python Machine Learning*, 3rd edition, Packt Publishing, 2019.
3. T. M. Mitchell, *Machine Learning*, 1st edition, McGraw Hill Education, 2017.
4. C. M Bishop, *Pattern Recognition and Machine Learnin*g, 1st edition, Springer, 2016.

CSC-401: Advanced Data Structures Using C
(Contact Hours: 75, Credits: 3 + 1=4)

(Marks: 56(T)+19(P)=75 (External) + 25 (Internal)=100)

Course Objectives

- CO1: To understand the fundamental concepts of data structures and their applications in problem-solving.
- CO2: To gain proficiency in implementing various types of linked lists, trees, and graphs using the C programming language.
- CO3: To analyse the time and space complexities of different data structures and their operations.
- CO4: To develop problem-solving skills through the application of data structures in real-world scenarios.
- CO5: To foster the ability to design efficient algorithms by leveraging appropriate data structures.

Learning Outcomes:

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

- LO1: Explain the importance of data structures in computer science and software engineering.
- LO2: Implement and manipulate singly linked lists, doubly linked lists, and circular linked lists in C.
- LO3: Construct and traverse binary trees, binary search trees, AVL trees, and B-trees using C.
- LO4: Create and manipulate various types of graphs including directed graphs, undirected graphs, weighted graphs in C.
- LO5: Apply appropriate data structures to efficiently solve problems in areas such as searching, and graph traversal.
- LO6: Design and implement algorithms that utilize data structures to optimize performance and memory usage.
- LO7: Demonstrate proficiency in writing clean, well-documented, and efficient C code for implementing data structures and algorithms.
- LO8: Understand the use of data structures in everyday applications theoretically.

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Linked Lists and Hashing	15	18	19
II	Trees	15	19	
III	Graphs and Applications of Data Structures	15	19	

IV	Practical	30	19	6
Total		**75**	**75**	**25**

Detailed Syllabus

Unit I: Linked Lists and Hashing **15 Hours**

Review of Linked Lists: Singly Linked List operations (insertion, search, deletion of a node), Implementation of Doubly and Circular Linked Lists: insertion, search, deletion of a node, Implementation of Circular and Priority Queue using linked lists. Hashing, Hash Functions: division method, mid square method, folding; analysis of ideal hash function; Conflict resolution techniques: linear and quadratic probing, double hashing, separate chaining; analysis of collision resolution techniques.

Unit II: Trees **15 Hours**

Tree as a data structure, Terminology: father, son, descendant, ancestor, height, depth, leaf node, forest, ordered trees, strictly binary tree, complete binary tree, internal nodes, external nodes; Representation of trees using linked lists, Tree traversal methods: pre-order, in-order, post-order; recursive algorithms for traversal methods, Binary Search Trees: creation, insertion, traversal, deletion of a node; AVL Trees: balance factor, tree rotation (single and double rotation), node insertion and tree traversal; Threaded binary tree: node insertion and tree traversal; B-tree and B+ tree: characteristics, node insertion, node search and tree traversal.

Unit III: Graphs and Applications of Data Structures **15 Hours**
Definition of a graph, Terminology: vertex, arc, directed, undirected, cardinality, finite and infinite graph, incidence, adjacency, indegree, outdegree, path length, weighted graph, connected graph, cyclic and acyclic graph, symmetric graph, complete graph, sub-graph; Graph representation: Adjacency matrix, adjacency lists, incidence matrix; Traversal schemes: recursive algorithms for Depth First Search (directed and undirected graphs), Breadth First Search (directed and undirected graphs); Shortest Path algorithm: Dijkstra’s algorithm for undirected weighted graphs.

Applications of Data Structures: Link Analysis: History of Search Engines, PageRank (definition and computation). Social Networks as a graph, Types of Social Network, Direct Discovery of communities in a social graph (Clique Percolation Method (CPM) algorithm). Introduction to Recommendation Systems, Types of Recommendation Systems paradigms, Collaborative-Filtering System paradigm using Nearest-Neighbor Technique.

Unit IV: Practical **30 Hours**

List of Practical Assignments (Questions may not be restricted to this list)

1. Write a program to perform various operations (insertion at beginning, insertion at end, insertion in the middle, deletion, searching, merging) on a doubly linked list.
2. Write a program to perform various operations (insertion at beginning, insertion at end, insertion in the middle, deletion, searching, merging, reversing) on a circular linked list.
3. Write programs to implement circular queues and priority queues using linked list representation, with operations for adding, searching and deleting values.

4. Write a menu driven program for a Binary Search Tree having functions for inserting, searching and deleting nodes. Also have recursive functions to traverse the tree in preorder, inorder and postorder techniques.
5. Write a program to implement an AVL tree with functions for inserting a node, searching a node and displaying (traversing) all nodes in the tree.
6. Write a program to implement a B-tree of order ‘m’ with functions for inserting values and for displaying all the nodes in the tree.
7. Write a program to implement a hash table of size ’n’. Insert values into appropriate table locations using h(k) = k MOD n. If a collision occurs, use linear probing to resolve it.
8. Write a program to implement a graph (directed/undirected) using adjacency matrix representation.
9. Write a program to implement a graph (directed/undirected) using adjacency list representation.
10. Write a program to implement Breadth First Search on directed and undirected graphs.
11. Write a program to implement Depth First Search on directed and undirected graphs.
12. Write a program to implement Dijkstra’s algorithm for an undirected weighted graph.

Instructions to Paper Setter (Theory)

Questions should be set according to the following scheme:

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
Total	**6**	**3**

Instructions to Paper Setter (Practical)

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered	Marks
I	2	1	7
II	3	2	6+6=12
III			
Total	**5**	**3**	**19**

Exam Duration

Theory	3 Hours
Practical	3 Hours

Suggested Readings

Texts:

1. S. Chattopadhyay, D. G. Dastidar, M. Chattopdhyay, *Data Structures Through C Language*, 1st Edition, BPB Publications, 2001.
2. Reema Thareja, *Data Structures Using C*, 3rd Edition, Oxford University Press, India, 2023.
3. R. Shankarmani, M. Vijayalakshmi, *Big Data Analytics*, 2/e, Wiley, 2016.

References:

1. N. Karumanchi, *Data Structures and Algorithms Made Easy*, 5th Edition, CareerMonk Publications, India, 2016.
2. Y. Kanetkar, *Data Structures Through C Language*, 5th Edition, BPB Publications, 2023.
3. S. K. Srivastava, D. Srivastava, *Data Structures Through C In Depth*, 2nd Edition, BPB Publications, 2021.

CSC-402: Compiler Design
(Contact Hours: 60, Credits: 4)

(Marks: 75 (External) + 25 (Internal)=100)

Course Objectives

- CO1: Understand the principles and techniques of compiler design.
- CO2: Learn about lexical analysis, parsing, semantic analysis, optimization, and code generation.
- CO3: Understand the role of compilers in software development and their impact on performance.

Learning Outcomes

- LO1: Demonstrate proficiency in designing and implementing lexical analyzers, parsers, and semantic analyzers for programming languages.
- LO2: Analyze and evaluate different parsing algorithms and their suitability for specific language grammars.
- LO3: Evaluate and compare the performance of different compiler optimizations and code generation strategies.
- LO4: Apply theoretical concepts to real-world compiler development projects.
- LO5: Communicate effectively about compiler design principles and techniques through written reports and oral presentations.

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Introduction to Compiler Design and Lexical Analysis	15	18	25
II	Syntax Analysis, SDDs and Intermediate Code Generation	15	19	
III	Type Checking, Symbol Tables and Run-Time Environments	15	19	
IV	Code Generation and Optimization	15	19	
Total		**60**	**75**	**25**

Detailed Syllabus

Unit I: Introduction to Compiler Design and Lexical Analysis 15 Hours

Language Processors, structure of a compiler, programming language basics-static/dynamic distinction, environments and states, static scope and block structure, explicit access control, dynamic scope, parameter passing mechanisms, aliasing;

Lexical Analysis- Role of the Lexical Analyzer, Specification of Tokens, Recognition of Tokens

Unit II: Syntax Analysis, SDDs and Intermediate Code Generation 15 Hours

Syntax Analysis- Role of the Parser, Error handling, Context-Free Grammars, Left Recursion, recursive decent parsing, shift-reduce parsing, LR parsing algorithm

Syntax-directed Translation**-** Syntax-Directed Definitions (SDDs).

Intermediate Code Generation - Three-Address Code, Quadruples and Triples, Three-Address Code Generation for assignment and Boolean expressions.

Unit III: Type Checking, Symbol Tables and Run-Time Environments 15 Hours

Type Checking-Type Expressions, Type checking of expressions and statements, type equivalence
Symbol table- Contents and features of a symbol table, Data structures for symbol table-List, Trees and hash tables;

Run-Time Environments –Static vs. Dynamic Storage Allocation, Activation records – environment without local procedures, environment with local procedures, Display;

Unit IV: Code Generation and Optimization 15 Hours

Code generation- Factors affecting code generation, Basic Block, Representation of basic Blocks, Code Generation for trees.

Code Optimization- need of optimization, problems in optimizing, Techniques for optimizing-compile time evaluation, common sub-expression elimination, variable propagation, code movement optimization, strength reduction, dead code elimination, loop optimization;

Instructions to Paper Setter

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
IV	2	1
Total	**8**	**4**

Exam Duration

Theory	3 Hours
Practical	-

Suggested Readings

Texts:

1. Chattopadhyay, S**.,** *Compiler Design*, 1st Edition, PHI Learning Pvt. Ltd, 2009
2. A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman,*Compilers Principles, Techniques and Tools,* 2nd Edition,Pearson Education, Fifth Impression, 2011

References:

1. K. Louden, *Compiler Construction: Principles and Practice*, 1st Edition, Cengage Learning, 1997
2. K. Muneeswaran**,** *Compiler Design*, 1st Edition, Oxford University Press, 2013

CSC-403: Introduction to Data Science using Python
(Contact Hours: 75, Credits: 3+1=4)

(Marks: 56(T)+19(P)=75 (External) + 25 (Internal)=100)

Course Objectives

The main objectives of this course are:

- CO1: Develop proficiency in Python programming language, including data manipulation, visualization, and analysis using libraries such as Pandas, and Matplotlib.
- CO2: Gain a solid understanding of foundational data science concepts and techniques, including data wrangling, exploratory data analysis
- CO3: Apply data science techniques to real-world datasets through hands-on projects and exercises, gaining practical experience in data cleaning and visualization

Learning Outcomes

- LO1: Recall and explain key functions and methods from Python libraries such as Pandas, and Matplotlib for data manipulation, analysis, and visualization.
- LO2: Demonstrate understanding of basic data science concepts, including data types, data cleaning techniques, and exploratory data analysis methods.
- LO3: Interpret summary statistics, data visualizations, and basic statistical analyses to gain
- LO4: Apply Python programming skills to manipulate, clean, and preprocess raw datasets for further analysis.
- LO5: Develop data-driven insights and communicate findings effectively through data visualization, interpretation, and presentation.

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Introduction to Data Science and pandas Library	15	19	19
II	Data Pre-processing, Transformation and Wrangling	15	19	
III	Data Visualization	15	18	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

Detailed Syllabus

Unit I : Introduction to Data Science and pandas Library 15 Hours

Introduction to Data Science: Need for Data Science, What is Data Science, Data Science Process, Business Intelligence and Data Science, Prerequisites for a Data Scientist, Components of Data Science, Tools and Skills needed, Issues of Ethics, Bias, and Privacy in Data Science; Data Analysis - Descriptive analysis – variable types, frequency distribution, measures of centrality, dispersion of a distribution; Diagnostic Analytics, Predictive Analytics, Prescriptive Analytics

Starting with pandas – Introduction to Pandas; The Series Data Structure – creating a series from Lists and Dictionaries working with indexes of a series; Querying a Series, DataFrame, DataFrame Indexing and Loading, Querying a DataFrame, Indexing Dataframes, Unique Values, Value Counts, and Membership

Data loading, storage and File formats – Reading and Writing data in Text format, reading text files in pieces, writing data to text format, working with CSV, JSON, Web Scraping, XML format, Interacting with web APIs, Reading from Excel files;

Unit II: Data Pre-processing, Transformation and Wrangling 15 Hours

Data Pre-processing – Data cleaning (Data wrangling, handling missing data, smoothing noisy data), Data Integration, Data Transformation, Data Reduction, Data Discretization;

Implementation in Python: Handling Missing Values – Finding Missing Data, Filtering out Missing Data, Filling in Missing Data; Summarizing and Computing descriptive statistics (count, describe, min, max, sum, mean, median, var, std, skew , kurt, cumsum, cummin, cummax), Correlation and Covariance

Data Transformation – Removing duplicates, Replacing values, Discretization and Binning, Detecting and Filtering outliers

Combining and Merging Datasets- merge, concat, combine_first

Data Wrangling -Join, Combine, Reshape: Hierarchical Indexing, Reordering and sorting levels, summary statistics by levels, Indexing with a Dataframe’s column, Reshaping with Hierarchical Indexing – stack, unstack;

Unit III: Data Visualization 15 Hours

Visualization: Why Visualization? Types of charts - Bar chart, Area chart, Line chart, Scatter plot, Box-whisker plot, Pie chart, Histogram, Heat map

Introduction to Matplotlib – Figures and Subplots, Colors, Markers and Line styles, Ticks, Labels and Legends, Annotations and Drawing on a Subplot, Saving plots to files; Plotting with pandas and Seaborn - Line plots, Bar plots, Histogram and Density plots, Scatter and Point plots, Heatmap, Facet Grids and Categorical Data;

Unit IV :Practical 30 Hours

List of practical assignments. (Problems may not be restricted to this list)

1. Create a Series from a list of temperatures and their corresponding days of the week. Perform operations to find the average temperature, the maximum temperature, and the day with the minimum temperature.
2. Given a DataFrame containing student information (e.g., names, ages, grades), index the DataFrame by student names and perform queries to retrieve information about specific students (e.g., find the grade of a particular student).
3. Load a CSV file containing sales data into a DataFrame. Perform data cleaning tasks such as handling missing values and removing duplicates. Save the cleaned data to a new CSV file.
4. Read a text file containing customer reviews, and extract relevant information such as product names, ratings, and review text. Save the extracted data to a new CSV file.
5. Retrieve data from a public API (e.g., weather data API) using Python. Process the retrieved JSON data and create a DataFrame with relevant information. Additionally, read data from an Excel file containing sales data and merge it with the API data to perform analysis.
6. Load a dataset containing missing values and identify the locations of missing data. Implement methods to filter out rows with missing data and fill in missing values using appropriate techniques such as mean imputation or interpolation.
7. Load a dataset and compute descriptive statistics such as count, mean, median, variance, standard deviation, skewness, kurtosis, etc. Explore the correlation between different numerical variables in the dataset.
8. Preprocess a dataset by removing duplicate rows, replacing specific values with others, and discretizing numerical variables into bins. Detect outliers in the dataset and filter them out.
9. Load multiple datasets containing related information and merge them using different methods such as concatenation, merging, or combining. Handle missing values and duplicate rows appropriately during the merging process.
10. Perform data wrangling tasks such as joining datasets based on common keys, reshaping datasets using hierarchical indexing, and summarizing statistics by levels of the index. Reorder and sort levels of the hierarchical index as needed.
11. Create a figure with multiple subplots arranged in a grid layout. Customize each subplot with different data, labels, and styles. Adjust the spacing between subplots and add overall titles.
12. Plot a dataset using Matplotlib, customizing colors, markers, line styles, and tick marks. Add gridlines and annotations to highlight specific data points or regions of interest.
13. Create a plot with labeled axes, a legend explaining different data series, and annotations to highlight important features. Customize the legend position, font size, and style.
14. Generate a plot and save it to a file in different formats such as PNG, JPEG, PDF, or SVG. Specify the resolution and quality settings for the saved image.

Instructions to Paper Setter (Theory)

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
Total	**6**	**3**

Instructions to Paper Setter (Practical)

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered	Marks
I	3	2	7+7=14
II			
III	2	1	5
Total	**5**	**3**	**19**

Exam Duration

Theory	3 Hours
Practical	3 Hours

Suggested Readings:

Texts:

1. B. Uma Maheswari, R. Sujatha, *Introduction to Data Science: Practical Approach with R and Python*, 1st Edition, Wiley India, 2021
2. Wes McKinney, *Python for Data Analysis, Data Wrangling with Pandas, NumPy and IPython*, 3rd Edition, Shroff/O'Reilly, 2022

References:

1. Sanjeev J. Wagh, Manisha S. Bhende, Anuradha D. Thakare, *Fundamentals of Data Science*, 1st Edition, Chapman/CRC, 2021
2. John Paul Mueller, Luca Massaron, *Python for Data Science For Dummies*, 3rd Edition, John Wiley & Sons, 2023
3. Rachel Schutt, Cathy O'Neil, *Doing Data Science: Straight Talk from the Frontline*, 1st Edition, Schroff/O'Reilly, 2013.
4. Foster Provost, Tom Fawcett, *Data Science for Business What You Need to Know*
5. *About Data Mining and Data-Analytic Thinking*, 1st Edition, O'Reilly, 2013.

CSC-404: Introduction to Data Science using Python
(Contact Hours: 75, Credits: 3+1=4)

(Marks: 56(T)+19(P)=75 (External) + 25 (Internal)=100)

Course Objectives

The main objectives of this course are:

- CO1: Develop proficiency in Python programming language, including data manipulation, visualization, and analysis using libraries such as Pandas, and Matplotlib.
- CO2: Gain a solid understanding of foundational data science concepts and techniques, including data wrangling, exploratory data analysis
- CO3: Apply data science techniques to real-world datasets through hands-on projects and exercises, gaining practical experience in data cleaning and visualization

Learning Outcomes

- LO1: Recall and explain key functions and methods from Python libraries such as Pandas, and Matplotlib for data manipulation, analysis, and visualization.
- LO2: Demonstrate understanding of basic data science concepts, including data types, data cleaning techniques, and exploratory data analysis methods.
- LO3: Interpret summary statistics, data visualizations, and basic statistical analyses to gain
- LO4: Apply Python programming skills to manipulate, clean, and preprocess raw datasets for further analysis.
- LO5: Develop data-driven insights and communicate findings effectively through data visualization, interpretation, and presentation.

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Introduction to Data Science and pandas Library	15	19	19
II	Data Pre-processing, Transformation and Wrangling	15	19	
III	Data Visualization	15	18	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

Detailed Syllabus

Unit I : Introduction to Data Science and pandas Library **15 Hours**

Introduction to Data Science: Need for Data Science, What is Data Science, Data Science Process, Business Intelligence and Data Science, Prerequisites for a Data Scientist, Components of Data Science, Tools and Skills needed, Issues of Ethics, Bias, and Privacy in Data Science; Data Analysis - Descriptive analysis – variable types, frequency distribution, measures of centrality, dispersion of a distribution; Diagnostic Analytics, Predictive Analytics, Prescriptive Analytics

Starting with pandas – Introduction to Pandas; The Series Data Structure – creating a series from Lists and Dictionaries working with indexes of a series; Querying a Series, DataFrame, DataFrame Indexing and Loading, Querying a DataFrame, Indexing Dataframes, Unique Values, Value Counts, and Membership

Data loading, storage and File formats – Reading and Writing data in Text format, reading text files in pieces, writing data to text format, working with CSV, JSON, Web Scraping, XML format, Interacting with web APIs, Reading from Excel files;

Unit II: Data Pre-processing, Transformation and Wrangling **15 Hours**

Data Pre-processing – Data cleaning (Data wrangling, handling missing data, smoothing noisy data), Data Integration, Data Transformation, Data Reduction, Data Discretization;

Implementation in Python: Handling Missing Values – Finding Missing Data, Filtering out Missing Data, Filling in Missing Data; Summarizing and Computing descriptive statistics (count, describe, min, max, sum, mean, median, var, std, skew , kurt, cumsum, cummin, cummax), Correlation and Covariance

Data Transformation – Removing duplicates, Replacing values, Discretization and Binning, Detecting and Filtering outliers

Combining and Merging Datasets- merge, concat, combine_first

Data Wrangling -Join, Combine, Reshape: Hierarchical Indexing, Reordering and sorting levels, summary statistics by levels, Indexing with a Dataframe's column, Reshaping with Hierarchical Indexing – stack, unstack;

Unit III: Data Visualization **15 Hours**

Visualization: Why Visualization? Types of charts - Bar chart, Area chart, Line chart, Scatter plot, Box-whisker plot, Pie chart, Histogram, Heat map

Introduction to Matplotlib – Figures and Subplots, Colors, Markers and Line styles, Ticks, Labels and Legends, Annotations and Drawing on a Subplot, Saving plots to files; Plotting with pandas and Seaborn - Line plots, Bar plots, Histogram and Density plots, Scatter and Point plots, Heatmap, Facet Grids and Categorical Data;

Unit IV :Practical **30 Hours**

List of practical assignments. (Problems may not be restricted to this list)

15. Create a Series from a list of temperatures and their corresponding days of the week. Perform operations to find the average temperature, the maximum temperature, and the day with the minimum temperature.
16. Given a DataFrame containing student information (e.g., names, ages, grades), index the DataFrame by student names and perform queries to retrieve information about specific students (e.g., find the grade of a particular student).
17. Load a CSV file containing sales data into a DataFrame. Perform data cleaning tasks such as handling missing values and removing duplicates. Save the cleaned data to a new CSV file.
18. Read a text file containing customer reviews, and extract relevant information such as product names, ratings, and review text. Save the extracted data to a new CSV file.
19. Retrieve data from a public API (e.g., weather data API) using Python. Process the retrieved JSON data and create a DataFrame with relevant information. Additionally, read data from an Excel file containing sales data and merge it with the API data to perform analysis.
20. Load a dataset containing missing values and identify the locations of missing data. Implement methods to filter out rows with missing data and fill in missing values using appropriate techniques such as mean imputation or interpolation.
21. Load a dataset and compute descriptive statistics such as count, mean, median, variance, standard deviation, skewness, kurtosis, etc. Explore the correlation between different numerical variables in the dataset.
22. Preprocess a dataset by removing duplicate rows, replacing specific values with others, and discretizing numerical variables into bins. Detect outliers in the dataset and filter them out.
23. Load multiple datasets containing related information and merge them using different methods such as concatenation, merging, or combining. Handle missing values and duplicate rows appropriately during the merging process.
24. Perform data wrangling tasks such as joining datasets based on common keys, reshaping datasets using hierarchical indexing, and summarizing statistics by levels of the index. Reorder and sort levels of the hierarchical index as needed.
25. Create a figure with multiple subplots arranged in a grid layout. Customize each subplot with different data, labels, and styles. Adjust the spacing between subplots and add overall titles.
26. Plot a dataset using Matplotlib, customizing colors, markers, line styles, and tick marks. Add gridlines and annotations to highlight specific data points or regions of interest.
27. Create a plot with labeled axes, a legend explaining different data series, and annotations to highlight important features. Customize the legend position, font size, and style.
28. Generate a plot and save it to a file in different formats such as PNG, JPEG, PDF, or SVG. Specify the resolution and quality settings for the saved image.

Instructions to Paper Setter (Theory)

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
Total	**6**	**3**

Instructions to Paper Setter (Practical)

Questions should be set according to the following scheme.

Unit	**Questions to be set**	**Questions to be answered**	**Marks**
I	3	2	7+7=14
II			
III	2	1	5
Total	**5**	**3**	**19**

Exam Duration

Theory	3 Hours
Practical	3 Hours

Suggested Readings:

Texts:

6. B. Uma Maheswari, R. Sujatha, *Introduction to Data Science: Practical Approach with R and Python*, 1st Edition, Wiley India, 2021
7. Wes McKinney, *Python for Data Analysis, Data Wrangling with Pandas, NumPy and IPython*, 3rd Edition, Shroff/O'Reilly, 2022

References:

8. Sanjeev J. Wagh, Manisha S. Bhende, Anuradha D. Thakare, *Fundamentals of Data Science*, 1st Edition, Chapman/CRC, 2021
9. John Paul Mueller, Luca Massaron, *Python for Data Science For Dummies*, 3rd Edition, John Wiley & Sons, 2023
10. Rachel Schutt, Cathy O'Neil, *Doing Data Science: Straight Talk from the Frontline*, 1st Edition, Schroff/O'Reilly, 2013.
11. Foster Provost, Tom Fawcett, *Data Science for Business What You Need to Know*
12. *About Data Mining and Data-Analytic Thinking*, 1st Edition, O'Reilly, 2013.

CSC-450: Mobile App Development
(Contact Hours: 75, Credits: 3+1=4)

(Marks: 56(T)+19(P)=75 (External) + 25 (Internal)=100)

Course Objectives

The main objectives of this course are:

- CO1: Introducemobileapplicationsdevelopmenttothestudentsusingtheandroidplatformasthebasisofteachingthetechniquesand
- CO2: IntroduceUI/UXapplicationconcepts.
- CO3: IntroduceDesignpatternsandequippingstudentswiththeessentialknowledgeandskillstodesign, develop,anddeployAndroidapplications.

Learning Outcomes

- LO1: Describe and define android programming structure
- LO2: Extend ones knowledge in application development
- LO3: Obtain new findings through experiment and demonstration
- LO4: Design and develop their own application which is tailored to their needs
- LO5: Create and formulate plans and solutions that will help solve problems
- LO6: Integrate the skills with the ability to access and decide self-selected criteria based on observation.

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Introduction to Android and Kotlin	10	14	19
II	Kotlin: Basic Programming.	17	21	
III	Android Layout, Fragments, and Databases	18	21	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

Detailed Syllabus

Unit-I: Introduction to Android and Kotlin **10 Hours**

Getting Started with Android and Kotlin: Why use Kotlin and Android? Setting up Android Studio, The structure of Android's code, Writing first Kotlin code, Deploying the app. Kotlin, XML, and the UI Designer: Exploring the project's Kotlin code and the main layout's XML code, Adding Text View, Edit Text; Button: Button, Radio Button And Radio Group; Checkbox to the main layout file. Exploring Android Studio and the Project Structure, Exploring the Android emulator.

Getting Started with Layouts and Material Design: Exploring Android UI design. Layouts, Creating the Exploring Layouts project, Building a menu with LinearLayout, Wiring up the UI with the Kotlin code, Adding layouts within layouts, Building a precise UI with ConstraintLayout, Laying out data with TableLayout. Building a UI with CardView and ScrollView. The Android Lifecycle: The life and times of an Android app, How to handle the lifecycle phases

Unit-II: Kotlin: Basic Programming. **17 Hours**

Kotlin: Variables, Operators, and Expressions, Decisions and Loop, while loops, do-while loops, Ranges, For loops, Controlling loops with break and continue, Functions. Introducing Object-*Oriented Programming(OOP):* Basic classes, Visibility modifiers, Constructors, Basic classes app and using the init block. Inheritance in Kotlin, Using inheritance with open classes, polymorphism, Classes using the Inheritance example app. Connecting Kotlin to the UI and Nullability, Kotlin interfaces, Using buttons and TextView widgets from our layout, Nullability – val and var revisited.

Unit-III: Android Layout, Fragments, and Databases. **18 Hours**

Bringing Android Widgets to Life: Declaring and initializing the objects from the layout, Creating UI widgets from pure Kotlin without XML, Exploring the palette, Lambdas, The widget exploration app, Coding the widget exploration app, Running the Widget Exploration app, Converting layouts to ConstraintLayout. Android Dialog Windows. Adapters and Recyclers: Adding RecyclerView, RecyclerAdapter, and ArrayList to the Note to Self project. Data Persistence and Sharing: The Android Intent class, Adding a settings page to Note to self, Persisting data with SharedPreferences, Reloading data with SharedPreferences, Making the Note to self settings persist. Design Patterns, Multiple Layouts, and Fragments: Introducing the model-view-controller pattern, Real-world apps, Device detection mini app, Configuration qualifiers. First fragment app. Android Databases: The SQL syntax primer, The Android SQLite API, Coding the database class, Coding the Fragment classes to use the DataManagerclass,Running the Age Database app.

Unit-IV: Practical **30 Hours**

List of practical assignments. (Problems may not be restricted to this list)

1. Write a Kotlin program to display a welcome message.
2. Write a Kotlin program to accept enrolment number, student name and marks of 5 subjects, from the user. Calculate the Total and Percentage and display all the details of the student on the screen.
3. Write a Kotlin program to convert Kilometre to Metres or vice versa. Accept the choice from

the user and perform the conversion accordingly. Note: Try to use lambda function for conversion.

4. Write a menu driven Kotlin program to provide a list of options to the user for finding the area of Circle, Square, Triangle and Cylinder. Perform appropriate operation as selected by the user.
5. Program for building a simple user interface using an XML for UI layout.
6. Develop a program to implement linear layout and absolute layout.
7. Develop a program to implement Text View and Edit Text.
8. Develop a program to implement Auto Complete Text View.
9. Develop a program to implement Button, Radio Button, Radio Group.
10. Develop a program to implement login window using above UI controls.
11. Develop a program to implement Checkbox.
12. Develop a program to implement List View, Grid View, and Scroll View.
13. Develop a program to create an activity.
14. Develop a program to implement new activity using explicit intent and implicit intent.
15. Build a simple application to store and retrieve data from SQLite database using Kotlin.
16. Create sample application with login module. (Check username and password) On successful login, Change TextView "Login Successful". And on login fail, alert user using Toast "Login fail".
17. Create login application where you will have to validate username and password until the username and password is not validated, login button should remain disabled.
18. Develop a Simple Android Application to design a simple Navigation between two activities.
19. Develop a Simple Android Application to design a Calculator Application.
20. Develop an application that makes use of database.
21. Create an application that will display admin panel and student panel using fragment.
22. Create an application that will display objective question layout using radio button and when option submitted show message "correct".
23. Write an android application using SQLite to create table and perform CRUD operations (Example. COURSE table (ID, Name, Duration, and Description), perform ADD, UPDATE, DELETE and READ operations)

Instructions to Paper Setter (Theory)

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
Total	**6**	**3**

Instructions to Paper Setter (Practical)

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered	Marks
I	3	2	7+7=14

II			
III	2	1	5
Total	**5**	**3**	**19**

Exam Duration

Theory	3 Hours
Practical	3 Hours

Suggested Readings:

Texts:

1. John, Horton, *Android Programming with Kotlin for Beginners: Build Android apps starting from zero programming experience with the new Kotlin programming language*, 1st Edition, Packt Publishing Ltd, 2019.
2. Peter, Spath, *Learn Kotlin for Android Development: The Next Generation Language for Modern Android Apps Programming*, 1st Edition, Apress Berkeley, CA, 2019.

References:

1. Neil Smyth, *Android Studio 4.2 Development Essentials - Kotlin Edition: Developing Android applications using Android Studio 4.2, Kotlin, and Android Jetpack*, 1st Edition, Packt Publishing Ltd, 2021.
2. Peter Sommerhoff, *Kotlin for Android App Development*, 1st Edition, Addison-Wesley Professional, 2018.
3. Greg Lim, *Beginning Android Development With Kotlin*, 2nd Edition, Greg Lim, 2022.
4. Pierre-Olivier Laurence, Amanda Hinchman-Dominguez, G. Blake Meike, Mike Dunn. *Programming Android with Kotlin*, 1st Edition, O'Reilly Media, 2021.

CSC-451: Mobile App Development
(Contact Hours: 75, Credits: 3+1=4)

(Marks: 56(T)+19(P)=75 (External) + 25 (Internal)=100)

Course Objectives

The main objectives of this course are:

- CO1: Introducemobileapplicationsdevelopmenttothestudentsusingtheandroidplatformasthebasisofteachingthetechniquesand
- CO2: IntroduceUI/UXapplicationconcepts.
- CO3: IntroduceDesignpatternsandequippingstudentswiththeessentialknowledgeandskillstodesign, develop,anddeployAndroidapplications.

Learning Outcomes

- LO1: Describe and define android programming structure
- LO2: Extend ones knowledge in application development
- LO3: Obtain new findings through experiment and demonstration
- LO4: Design and develop their own application which is tailored to their needs
- LO5: Create and formulate plans and solutions that will help solve problems
- LO6: Integrate the skills with the ability to access and decide self-selected criteria based on observation.

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Introduction to Android and Kotlin	10	14	19
II	Kotlin: Basic Programming.	17	21	
III	Android Layout, Fragments, and Databases	18	21	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

Detailed Syllabus

Unit-I: Introduction to Android and Kotlin **10 Hours**

Getting Started with Android and Kotlin: Why use Kotlin and Android? Setting up Android Studio, The structure of Android's code, Writing first Kotlin code, Deploying the app. Kotlin, XML, and the UI Designer: Exploring the project's Kotlin code and the main layout's XML code, Adding Text View, Edit Text; Button: Button, Radio Button And Radio Group; Checkbox to the main layout file. Exploring Android Studio and the Project Structure, Exploring the Android emulator.

Getting Started with Layouts and Material Design: Exploring Android UI design. Layouts, Creating the Exploring Layouts project, Building a menu with LinearLayout, Wiring up the UI with the Kotlin code, Adding layouts within layouts, Building a precise UI with ConstraintLayout, Laying out data with TableLayout. Building a UI with CardView and ScrollView. The Android Lifecycle: The life and times of an Android app, How to handle the lifecycle phases

Unit-II: Kotlin: Basic Programming. **17 Hours**

Kotlin: Variables, Operators, and Expressions, Decisions and Loop, while loops, do-while loops, Ranges, For loops, Controlling loops with break and continue, Functions. Introducing Object-*Oriented Programming(OOP):* Basic classes, Visibility modifiers, Constructors, Basic classes app and using the init block. Inheritance in Kotlin, Using inheritance with open classes, polymorphism, Classes using the Inheritance example app. Connecting Kotlin to the UI and Nullability, Kotlin interfaces, Using buttons and TextView widgets from our layout, Nullability – val and var revisited.

Unit-III: Android Layout, Fragments, and Databases. **18 Hours**

Bringing Android Widgets to Life: Declaring and initializing the objects from the layout, Creating UI widgets from pure Kotlin without XML, Exploring the palette, Lambdas, The widget exploration app, Coding the widget exploration app, Running the Widget Exploration app, Converting layouts to ConstraintLayout. Android Dialog Windows. Adapters and Recyclers: Adding RecyclerView, RecyclerAdapter, and ArrayList to the Note to Self project. Data Persistence and Sharing: The Android Intent class, Adding a settings page to Note to self, Persisting data with SharedPreferences, Reloading data with SharedPreferences, Making the Note to self settings persist. Design Patterns, Multiple Layouts, and Fragments: Introducing the model-view-controller pattern, Real-world apps, Device detection mini app, Configuration qualifiers. First fragment app. Android Databases: The SQL syntax primer, The Android SQLite API, Coding the database class, Coding the Fragment classes to use the DataManagerclass,Running the Age Database app.

Unit-IV: Practical **30 Hours**

List of practical assignments. (Problems may not be restricted to this list)

24. Write a Kotlin program to display a welcome message.
25. Write a Kotlin program to accept enrolment number, student name and marks of 5 subjects, from the user. Calculate the Total and Percentage and display all the details of the student on the screen.
26. Write a Kotlin program to convert Kilometre to Metres or vice versa. Accept the choice from

the user and perform the conversion accordingly. Note: Try to use lambda function for conversion.

27. Write a menu driven Kotlin program to provide a list of options to the user for finding the area of Circle, Square, Triangle and Cylinder. Perform appropriate operation as selected by the user.
28. Program for building a simple user interface using an XML for UI layout.
29. Develop a program to implement linear layout and absolute layout.
30. Develop a program to implement Text View and Edit Text.
31. Develop a program to implement Auto Complete Text View.
32. Develop a program to implement Button, Radio Button, Radio Group.
33. Develop a program to implement login window using above UI controls.
34. Develop a program to implement Checkbox.
35. Develop a program to implement List View, Grid View, and Scroll View.
36. Develop a program to create an activity.
37. Develop a program to implement new activity using explicit intent and implicit intent.
38. Build a simple application to store and retrieve data from SQLite database using Kotlin.
39. Create sample application with login module. (Check username and password) On successful login, Change TextView "Login Successful". And on login fail, alert user using Toast "Login fail".
40. Create login application where you will have to validate username and password until the username and password is not validated, login button should remain disabled.
41. Develop a Simple Android Application to design a simple Navigation between two activities.
42. Develop a Simple Android Application to design a Calculator Application.
43. Develop an application that makes use of database.
44. Create an application that will display admin panel and student panel using fragment.
45. Create an application that will display objective question layout using radio button and when option submitted show message "correct".
46. Write an android application using SQLite to create table and perform CRUD operations (Example. COURSE table (ID, Name, Duration, and Description), perform ADD, UPDATE, DELETE and READ operations)

Instructions to Paper Setter (Theory)

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
Total	**6**	**3**

Instructions to Paper Setter (Practical)

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered	Marks
I	3	2	7+7=14

II			
III	2	1	5
Total	**5**	**3**	**19**

Exam Duration

Theory	3 Hours
Practical	3 Hours

Suggested Readings:

Texts:

5. John, Horton, *Android Programming with Kotlin for Beginners: Build Android apps starting from zero programming experience with the new Kotlin programming language*, 1st Edition, Packt Publishing Ltd, 2019.
6. Peter, Spath, *Learn Kotlin for Android Development: The Next Generation Language for Modern Android Apps Programming*, 1st Edition, Apress Berkeley, CA, 2019.

References:

7. Neil Smyth, *Android Studio 4.2 Development Essentials - Kotlin Edition: Developing Android applications using Android Studio 4.2, Kotlin, and Android Jetpack*, 1st Edition, Packt Publishing Ltd, 2021.
8. Peter Sommerhoff, *Kotlin for Android App Development*, 1st Edition, Addison-Wesley Professional, 2018.
9. Greg Lim, *Beginning Android Development With Kotlin*, 2nd Edition, Greg Lim, 2022.
10. Pierre-Olivier Laurence, Amanda Hinchman-Dominguez, G. Blake Meike, Mike Dunn. *Programming Android with Kotlin*, 1st Edition, O'Reilly Media, 2021.

CSC-453: Natural Language Processing
(Contact Hours: 75, Credits: 3+1= 4)

(Marks: 56(T)+19(P)=75 (External) + 25 (Internal)=100)

Course Objectives

- CO1: Grasp NLP Fundamentals: Understand core concepts like language, ambiguity, data, and different NLP generations. Analyze word representation techniques and their strengths/weaknesses.
- CO2: Master Part-of-Speech Tagging: Identify and address ambiguity in POS tagging. Implement and evaluate different tagging approaches using NLTK.
- CO3: Explore Deep Parsing Techniques: Grasp the linguistic concepts behind parsing (constituency & dependency). Implement and evaluate parsing algorithms (CYK, Dependency Parsing) using NLTK.
- CO4: Apply NLP in Real-World Tasks: Understand the challenges of ambiguity in various NLP applications (NER, MT, Sentiment Analysis, Q&A, Conversational AI, Summarization).

Learning Outcomes

- LO1: Define Natural Language Processing (NLP) and its applications.
- LO2: Explain the core concepts involved in NLP, including language, linguistics, ambiguity, grammar, probability, and data.
- LO3: Identify the fundamental concepts of Part-of-Speech (POS) tagging and its importance in NLP.
- LO4: Explain the various applications of NLP.
- LO5: Explain the role of machine learning algorithms in parsing techniques (e.g., CYK parsing, dependency parsing).
- LO6: Implement NLP tasks using Python and the Natural Language Toolkit (NLTK) library (e.g., tokenization, text manipulation, corpus exploration, tagging, and parsing).
- LO7: Analyze the strengths and weaknesses of different parsing algorithms.
- LO8: Design and implement a simple NLP application using the acquired knowledge and tools.

Outline of the course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Introduction to NLP and Part-Of-Speech Tagging	15	20	19
II	Deep Parsing	15	20	
III	Applications of NLP – An Overview	15	16	
IV	Practical	30	19	6

Total	**75**	**75**	**25**

Detailed Syllabus

Unit I: Introduction to NLP and Part-Of-Speech Tagging **15 Hours**

Introduction to NLP: Language and Linguistics, Ambiguity and Layers of NLP, Grammar, Probability, and Data, Generations of NLP;

Representation and NLP: Ambiguity and Representations; Generation-2: Discrete Representational Semantics (n-Gram Vectors, Caveats, Statistical Language Models); Generation-3: Dense Representations (Dense Representation of Words, Neural Language Models)

Part-of-Speech Tagging (POS):Introduction to POS; Ambiguity in POS and the *-al* rule, Table Look-Up-Based and Rule-based POS Tagging); Statistical POS Tagging (HMM Based Formulation of POS tagging, Viterbi Decoding for POS Tagging).

Unit II: Deep Parsing **15 Hours**

*Linguistics of Parsing:*Heads and Modifiers, Relationship between Constituency and Dependency Parsing, Phrase Structure Grammar Rules;

Algorithmics of Parsing: Machine Learning and Parsing; CYK Parsing; Dependency Parsing (Arguments and Adjuncts, Unlabelled Dependency Graph Construction, Dependency Relations, Projectivity).

Unit III: Applications of NLP – An Overview **15 Hours**

Named Entity Recognition (NER):Problem Formulation, Ambiguity in NER, Datasets;

*Machine Translation (MT):*Introduction, Ambiguity Resolution in MT, RBMT-EBMT-SMT-NMT, Vauquois Triangle, Neural Machine Translation (Encoder-Decoder);

*Sentiment Analysis:*Problem Statement, Ambiguity for Sentiment Analysis, Lexicons for Sentiment Analysis, Valence, Arousal, and Dominance, Wheels of Emotions, Manual and Automatic Creation of Lexicons;

Question Answering (Q and A): Problem Formulation, Ambiguity in Q and A, Dataset Creation;

Conversational AI: Problem Definition, Ambiguity Resolution in Conversational AI;

Summarization: Ambiguity in Text Summarization, Problem Definitions;

Unit IV: Practical **Hours: 30**

Language and tool: Python and the Natural Language Toolkit (NLTK)

List of practical assignments. (Problems may not be restricted to this list)

1. Tokenization of text into sentences.
2. Tokenization of sentences into words.
3. Computing with language (Frequency Distribution, Fine-grained selection of words, Collocations and Bigrams)
4. Accessing Text Corpora (Gutenberg Corpus, Brown Corpus, Reuters Corpus) and experimenting with at least one annotated Corpus
5. Automatic Tagging (Default Tagger)
6. Unigram Taggers
7. Separating the Training and Testing Data
8. Bigram taggers, Trigram taggers
9. Tagging Unknown words
10. Storing Taggers
11. Experimenting with NLTK's Shift-Reduce Parser and Chart parser
12. Experimenting with NLTK's Dependency Grammar and Projective Dependency Grammar

Instructions to Paper Setter (Theory)

Questions should be set according to the following scheme.

Unit	**Questions to be set**	**Questions to be answered**
I	2	1
II	2	1
III	2	1
Total	**6**	**3**

Instructions to Paper Setter (Practical)

Questions should be set according to the following scheme.

Unit	**Questions to be set**	**Questions to be answered**
IV (Practical)	5	3
	At least one question from Tokenization or Computing with Language or Corpora, one question from Tagging, and one question from Parsing	

Exam Duration

Theory	3 hours
Practical	3 hours

Suggested Readings

Texts:

1. P. Bhattacharyya, A. Joshi, *Natural Language Processing*, 1st Edition, Wiley, 2023.
2. S. Bird, E. Klein, E Loper. *Natural Language Processing with Python*, Web Edition, nltk.org/book/

References:

1. D. Jurafsky, J.H. Martin, *Speech and Language Processing, An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*, 2nd Edition, Pearson Education India, 2013
2. J. Allen, *Natural Language Understanding*, 2nd Edition, Pearson Education, 2002
3. D. Chopra, N. Joshi, I. Mathur, *Mastering Natural Language Processing with Python*, 1st edition, Packt Publishing, 2016.

CSC-454: Fundamentals of Blockchain
(Contact Hours: 75, Credits: 3+1=4)

(Marks: 56(T)+19(P)=75 (External) + 25 (Internal)=100)

Course Objectives

- CO1: Students are expected to grasp a comprehensive understanding of the fundamental principles of blockchain technology, providing them with a broad overview of its essential concepts.
- CO2: The course aims to acquaint students with the Ethereum protocol and Solidity programming language, serving as a foundational step toward their proficiency in developing applications and programming within blockchain environments.
- CO3: Through the course, students will explore various types of blockchains and consensus algorithms, enabling them to gain insights into the diversity within blockchain technology and its underlying mechanisms.

Learning Outcomes

- LO1: Recall and define key concepts of blockchain technology, including distributed ledger, consensus mechanisms, cryptography, and smart contracts.
- LO2: Demonstrate a holistic understanding of blockchain fundamentals by explaining the relationship between various components, mechanisms, and use cases within blockchain technology.
- LO3: Design and Develop blockchain applications with Ethereum and Solidity

Outline of the course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Introduction to Blockchain	15	18	19
II	Cryptography in Blockchain and Ethereum	15	19	
III	Smart Contract Development with Solidity and Ethereum	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

Detailed Syllabus

Unit I : Introduction to Blockchain **15 Hours**

Introduction to Blockchain - Blockchain and Decentralization, What Is Blockchain, Milestones in Blockchain Development, Features of Blockchain, Blockchain Types and their Characteristics – Public, Private, Federated, Hybrid, Blockchain Framework - Hardware/Infrastructure Layer, Data Layer, Network Layer, Consensus Layer, Application and Presentation Layer; Terminologies – Block, Block Structure, Ledger, Distributed, Proof of Work, Block Awards , Transactions and UTXOs; Consensus – Proof of Work, Proof of Stake, Delegated Proof of Stake, Byzantine Fault Tolerance; Implementation in Python – creating a block, creating a blockchain, creating a genesis block, adding a block to a blockchain, creating a simple PoW algorithm, tamper-proof validation

Unit II: Cryptography in Blockchain and Ethereum 15 Hours

Cryptography Primitives- Hash Function, Properties of Hash Functions, Hash Pointers and Data Structures, Tampering, Role of Hashes in Blockchain – Block Validation and Digital Signatures, Secure Hash Algorithm (SHA), Public Key Cryptography, Merkle Tree, Merkle Tree Creation, Role of Merkle Tree in Blockchain; Public Key Cryptography, Public and Private Keys, Public Key Encryption Algorithms, Digitally Signed Transactions, Digital Signing in Blockchain
Ethereum: Overview of Ethereum Blockchain, Key Features, EVM, Ethereum Network, Smart Contracts, Challenges in Implementing Smart Contracts, Smart Contract Life Cycle, Ethereum Transactions, Transaction Life Cycle, Gas and Transaction Fees

Unit III :Smart Contract Development with Solidity and Ethereum 15 Hours

Solidity Fundamentals - Solidity Comments, Solidity Program and Version Declaration, Import a Solidity File, Constructor Function, Function Modifier, Blockchain Access Scope: Pure/View/Payable Functions, Function Access Scope: Public, External, Internal, and Private, Solidity Data Types – Boolean, Integer, Address, Byte Array, Fixed Sized Array, Dynamic Sized Array, Mapping Data Type, Enum Data Type Struct Data Type; Events – Ethereum events, Event storage, defining an event, emitting an event;
*Smart Contract Development with Solidity:*Setting up Ethereum development tools - Installing an Ethereum client, MetaMask, node.js, Truffle suite; Developing Smart Contracts – setting up a Truffle project, compiling and deploying a contract, Interacting with Smart Contracts through Web3, Creating DApp user interface for interacting with Smart Contracts.

Unit IV: Practical 30 Hours

List of practical assignments. (Problems may not be restricted to this list)

1. Create a Block class is defined to represent a block in a blockchain. Each block has the following attributes:

- data: The data or payload that the block contains.
- previous_hash: The hash of the previous block in the blockchain.
- nonce: A value that is incremented during the mining process.
- hash: The hash of the current block.

The Block class should have the following methods

- calculate_hash method to calculate the SHA256 hash of a block. It should concatenate the data, previous_hash,and nonce and then encodes and hashes the resulting string using the SHA256 algorithm
- mine_block method which takes the difficulty level as argument to implement a simple PoW algorithm. It increments the nonce value and recalculates the hash until the hash of

the block satisfies the difficulty requirement. The difficulty requirement is defined as a certain number of leading zeros in a hash.

Create a Blockchain class to represent the entire blockchain. It has the following attributes:

- chain: A list that holds all the blocks in a blockchain.
- difficulty: The difficulty level for mining new blocks; it specifies the number of leading zeros required in a block's hash.

The Blockchain class should have the following methods

- create_genesis_block method to create the first block in a blockchain, often called the genesis block. It is a special block with no previous hash.
- get_last_block method which returns the last block in a blockchain.
- add_block method which adds a new block to a blockchain. It takes a new_block as input, sets the previous hash of the new block to the hash of the last block in the chain, and then initiates the mining process by calling mine_block.

Create a Blockchain instance named blockchain and add few blocks by calling the add_block method, each with its own data. Finally, print the contents of the blockchain by iterating over each block in the chain and displaying the block's data and hash.

2. Create a Block class is defined to represent a block in the blockchain. Each block has the following attributes:
   - block_number: Number assigned to block.
   - timestamp: Timestamp indicating when block was created.
   - transactions: Transactions or data included in block.
   - previous_hash: Hash of previous block in blockchain.
   - gas_limit and gas_used: Attributes related to gas in a blockchain
   - miner: Miner responsible for mining the block.
   - hash: Hash of current block, calculated using calculate_hash method.

   The Block class should have the following method
   - calculate_hash method in the Block class calculates the SHA256 hash of the block. It concatenates the various attributes of the block into a single string and then encodes and hashes the resulting string using the SHA256 algorithm from hashlib.

   Create a Blockchain class to represent the entire blockchain. It has a single attribute:
   - chain: A list that holds all the blocks in the blockchain.

   The Blockchain class should have the following methods

   - create_genesis_block method creates the genesis block, which is a special block with no previous hash. It returns a new Block instance with predefined values for the block number, data, previous hash, gas limit, gas used, and miner.
   - get_last_block method which returns the last block in a blockchain.
   - add_block method that adds a new block to the blockchain. It takes a new_block as input, sets the previous hash of the new block to the hash of the last block in the chain (self.chain[-1].hash), and then appends the new block to the chain.
   - print_block method prints the contents of a block, including its block number, timestamp, transactions, previous hash, gas limit, gas used, miner, and hash.
   - traverse_chain method traverses the blockchain and calls the print_block method for each block in the chain, effectively printing the contents of each block.

Create an instance of the Blockchain class named my_blockchain. Add three blocks to the blockchain using the add_block method. Each block has its own block number, transactions, gas limit, gas used, and miner.The previous hash for each block is initially set to an empty string. The traverse_chain method is called to print the contents of each block in the blockchain.

3. Create a UTXO class to represent an unspent transaction output. Each UTXO has the following attributes:
   - txid: Transaction ID that UTXO belongs to.
   - index: Index of output in transaction.
   - value: Value of output

   and a method
   - __str__ method to provide a string representation of the UTXO.

Create a Transaction class to represent a transaction. Each transaction has the following attributes:
   - inputs: List of UTXOs being spent as inputs.
   - outputs: List of new UTXOs created as outputs

   and methods
   - __str__ method to provide a string representation of a transaction
   - hash method to generate a hash for the transaction by concatenating the transaction IDs and indices of the input UTXOs and the values of the output UTXOs. It then calculates the SHA256 hash of the resulting string using hashlib.

   Create Sample UTXO and Transaction instances and print them using the print function. Create hashes for the transactions using the hash method and print them.

4. Create a Block class to represent a single block in the blockchain. Each block has several attributes:
   - timestamp: Stores the time when block was created using time.time() function.
   - data: Represents data or transaction that the block contains.
   - previous_hash: Stores hash of previous block in chain.
   - nonce: Number incremented during mining to find a valid hash.
   - hash: Stores hash of block itself.

   The Block class also has the following methods:
   - generate_hash(): This method generates the hash of a block by concatenating the block’s attributes and applying the SHA-256 hash function from the hashlib module.
   - mine_block(difficulty): This method performs the mining process by incrementing the nonce value until a hash is found that meets the difficulty criteria. The difficulty is the number of leading zeros required in the hash.

   Create a Blockchain class that represents the blockchain itself and manages blocks. It has the following attributes and methods:
   - chain: List that stores blocks in blockchain
   - difficulty: Represents the difficulty level of mining.

   The Blockchain class also uses the following methods:
   - create_genesis_block(): This method creates the first block in a blockchain, called the genesis block, with arbitrary data and a previous hash of "0".
   - get_latest_block(): This method returns the most recently added block in a chain.

- add_block(new_block): This method adds a new block to a chain. It sets the previous hash of new block to the hash of the latest block, mines the new block, and appends it to the chain.
- is_chain_valid(): This method checks the validity of a blockchain by verifying the integrity of each block. It iterates through the chain and compares the hash and previous hash values of each block.

Create an instance of the Blockchain class and add three blocks to the chain. After mining each block, it checks the validity of the blockchain using the is_chain_valid() method. Next, modify the data of the second block in the chain to "Tampered transaction". Recheck the validity of the blockchain to show that it detected the tampering. The output of the code should include information about the mining process and the validity of the blockchain after each step.

5. Programs to demonstrate the creation and deployment of smart contracts in Solidity
6. Programs to demonstrate DApps for interacting with Smart contracts
7. Programs to demonstrate the use of variables, arrays, struct, enum, functions, events in Solidity

Instructions to Paper Setter

Questions should be set according to the following scheme.

Unit	Questions to be set	Questions to be answered
I	2	1
II	2	1
III	2	1
IV	2	1
Total	**8**	**4**

Instructions to Paper Setter (Practical)

Questions should be set according to the following scheme.

Unit	**Questions to be set**	**Questions to be answered**	**Marks**
I	2	1	9
II			
III	3	2	5+5=10
Total	**5**	**3**	**19**

Exam Duration

Theory	3 Hours
Practical	3 Hours

Suggested Readings

Texts:

1. Ramchandra S. Mangrulkar, Pallavi V. Chavan, *Blockchain Essentials Core Concepts and Implementations*, 1st Edition, Apress, 2024
2. Kevin Solorio,Randall Kanna,David H. Hoover, *Hands-On Smart Contract Development with Solidity and Ethereum: From Fundamentals to Deployment,* 1st Edition, O'Reilly Media, 2019

References:

1. Imran Bashir, *Mastering Blockchain*, 2nd Edition, Packt, 2018
2. Chandramouli Subramanian, Asha A George, Abhilash K A, Meena Karthikeyan, *Blockchain Technology*, 1st Edition, University Press, 2020
3. Daniel Drescher, *Blockchain Basics A Non-Technical Introduction in 25 Steps,* 1st Edition, Apress, 201
4. Weijia Zhang, Tej Anand,*Blockchain and Ethereum Smart Contract Solution Development, Dapp Programming with Solidity,* 1st Edition, Apress, 2022
5. ArshdeepBhaga, Vijay Madisetti, *Blockchain Applications-A Hands-on Approach,* 1st Edition, VPT, 2018

CSC-455: Data Mining
(Contact Hours: 60, Credits: 4)

(Marks: 75 (External) + 25 (Internal)=100)

Course Objectives

- CO1: Understand the fundamental concepts and techniques of data mining.
- CO2: Learn various data preprocessing techniques, including cleaning, integration, transformation, and reduction.
- CO3: Explore different data mining algorithms for classification, clustering, association rule mining, and outlier detection.
- CO4: Understand the ethical and privacy considerations in data mining, including issues related to data ownership, security, and confidentiality.
- CO5: Learn how to evaluate the performance of data mining models and interpret their results effectively.
- CO6: Explore applications of data mining in various domains such as business, healthcare, finance, and social media.

Learning Outcomes

- LO1: Apply data preprocessing techniques to clean, integrate, transform, and reduce data for effective data mining.
- LO2: Implement and evaluate classification algorithms to predict categorical outcomes based on input data.
- LO3: Implement and evaluate clustering algorithms to discover natural groupings in data.
- LO4: Implement and evaluate association rule mining algorithms to discover patterns in large datasets.
- LO5: Interpret the results of data mining models and make informed decisions based on the insights gained.
- LO6: Communicate effectively about data mining concepts, techniques, and results through written reports and presentations.

Outline of the Course

Unit	Topic	Minimum Class Hours	ExternalMarks	Internal Marks
I	Introduction to Data Mining, Data Preprocessing	12	15	25
II	Mining Frequent Patterns, Association and Correlations	17	24	
III	Classification, Prediction, Clustering	18	24	
IV	Outlier Analysis; Application of Data Mining	10	12	

Total	**60**	**75**	**25**

Detailed Syllabus

Unit I: Introduction to Data Mining; Data Preprocessing 12 Hours

What motivated data mining? Why is it important? What is data mining? Data mining-on what kind of data? Data mining functionalities. Are all patterns interesting? Classifications of data mining systems. Data mining task primitives. Major issues in data mining.

Why process the data? *Descriptive Data Summarization*: Mesuring central tendency, Measuring the dispersion, Garphicdisplays. *Data Cleaning*: Missing values, Noisy data, Data cleaning as a process. *Data Integration and Transformation*: Data integration, Data transformation. *Data Reduction*: Data cube aggregation, Attribute subset selection, Dimensionality reduction and Numerosity reduction. Data discretization and concept hierarchy generation.

Unit II: Mining Frequent Patterns, Association and Correlations 17 Hours
Basic Concepts: Market basket analysis; Frequent itemset, Closed itemsets and association rules; Frequent pattern mining.*Efficient and Scalable Frequent Itemset Mining Methods*: The Apriori algorithm, Generation association rules from frequent itemsets, Improving the efficiency of Apriori, Mining frequent itemset without candidate generation, Mining frequent itemset using vertical data format, Mining closed frequent itemsets. *Mining Various Kinds of Association Rules*: Mining multilevel association rules, Mining multidimensional association rules, *From Association Mining to Correlation Analysis*: Strong rules are not necessarily interesting, From association analysis to correlation analysis. *Constraint-based Association Rule Mining*: Metarule-guided mining of association rules, Constraint pushing.

Unit III: Classification, Prediction, Clustering 18 Hours
*Classification and Prediction:*What is classification? What is prediction? *Classification by Decision Tree Induction*: Decision tree induction, Attribute selection measures, Tree pruning. *Prediction*: Linear regression, Nonlinear Regression.

Cluster Analysis: What is cluster analysis? *Types of Data*: Interval-scaled variables, Binary variables, Categorical, Ordinal, Ratio-scaled, Variables of mixed types and vector objects. Categorization of major clustering methods. *Partitioning Methods*: k-means and k-medoids, *Partitioning in Large Databases*: k-medoids to CLARANS. *Hierarchical Methods*: Agglomerative and divisive hierarchical clustering, BIRCH, *Density-based Methods*: DBSCAN, *Grid-based methods*: STING. *Model-based methods*: Expectation-maximization

Unit V: Outlier Analysis, Applications of Data Mining 10 Hours
Outlier analysis: Statistical distribution based outlier detection, Distance-based outlier detection, Density-based outlier detection, Deviation based outlier detection.

*Applications:*Data mining for financial data analysis, Data mining for retail industry, Data mining for telecommunication industry, Data mining biological data analysis, Data mining for scientific applications, Data mining for intrusion detection. *Data Mining for System Products*: How to choose a data mining system, Examples of commercial data mining systems.

Additional Themes: Theoretical foundations, Statistical data mining, Visual and audio data mining, Data mining and collaborative filtering. Social impacts of data mining. Future trends of data mining

Instructions to Paper Setter

Questions should be set according to the following scheme.

Unit	**Questions to be set**	**Questions to be answered**
I	2	1
II	2	1
III	2	1
IV	2	1
Total	**8**	**4**

Exam Duration

Theory	3 Hours
Practical	-

Suggested Readings:

Texts:

1. J Han and M Kamber,*Data Mining: Concepts and Techniques*, 3rd Edition. Morgan Kaufman Publisher/ Elsevier, 2011.

References:

1. D Hand, H Mannila, P Smyth, *Principles of Data Mining*, Prentice-Hall India, 2005
2. A K Pujari,*Data Mining Techniques*, University Press, 2008
3. P Tan, V Kumar, M Steinbach,*Introduction to Data Mining*. Pearson Education, 2007.