UNIVERSITY OF CALCUTTA

Notification No. CSR/81/2026

It is notified for information of all concerned that in terms of the provisions of Section 54 of the Calcutta University Act, 1979, (as amended), and, in the exercise of his powers under 9(6) of the said Act, the Vice-Chancellor has, by an order dated 15.07.2026, approved the revised complete Syllabus of **Computer Application (Core Vocational) under CCF, 2022.**

The above shall take effect from the Odd Semester Examinations, 2026, and onwards.

22/07/2026

SENATE HOUSE
Kolkata-700073
22.07.2026

Prof.(Dr.) Debasis Das
Registrar

University of Calcutta

4 - Years B.Sc. Degree program in Computer Application under credit framework (CCF).

Semester – I, II, III, IV, V, VI, VII and VIII

(Core-Vocational)

Structure of4-Year B.Sc. program in Computer Application (Core Vocational).

Semester	Paper Code	Theory Paper	Practical	Modality of Practical	SEC	Practical	Modality of Practical
I	**DSCC-1**	Computer fundamentals and Digital Logic **Credit-3**	Digital logic Circuit Lab **Credit-1**	Full Marks = 25 Experiment =15 Viva = 05 LNB = 05 * in house	Data visualization using spreadsheet. **Credit-2**	Data visualization using spreadsheet **Credit-2**	Full Marks = 50 Experiment = 30 Viva-Voce = 10 LNB = 10 * In house
II	**DSCC-2**	Problem Solving using C. **Credit-3**	Problem Solving using C Lab. **Credit-1**	Full Marks = 25 Experiment =15 Viva = 05 LNB = 05 * in house	Web Development. **Credit-2**	Web Development lab. **Credit-2**	Full Marks = 50 Experiment = 30 Viva-Voce = 10 LNB = 10 * in house
III	**DSCC-3**	Data Structure **Credit-3**	Data structure lab **Credit-1**	Full Marks = 25 Experiment =15 Viva = 05 LNB = 05 * in house	Mobile App Development **Credit-2**	Mobile App Development Lab **Credit-2**	Full Marks = 50 Experiment = 30 Viva-Voce = 10 LNB = 10 * in house`
	DSCC-4	Programing in Python. **Credit-3**	Python lab **Credit-1**				

Semester	Paper Code	Theory Paper	Credit	Practical/Lab	Credit	Modality of Practical
IV	**DSCC-5**	Multimedia and Its Application	3	Multimedia Lab	1	Full Marks = 25 Experiment =15 Viva = 05 LNB = 05 * inhouse
	DSCC-6	Computer Architecture & Organization	3	Computer Architecture & Organization Lab	1	
	DSCC-7	Database Management system	3	RDBMS Lab	1	
	DSCC-8	Object Oriented Programming	3	OOPs usingJava	1	

Semester	Paper Code	Theory Paper	Credit	Practical/Lab	Credit	Modality of Practical
V	DSCC-9	Data Communication and Networking	3	Computer Networking Lab	1	Full Marks = 25 Experiment =15 Viva-Voce = 05 LNB = 05 * inhouse
	DSCC-10	Introduction to Algorithms	3	Lab using C	1	
	DSCC-11	Operating System	3	Operating Systems Lab	1	
	DSCC-12	Mathematical Methods	3	Lab using Python	1	
VI	DSCC-13	Introduction to Machine learning	3	ML Lab	1	Full Marks = 25 Experiment =15 Viva-Voce = 05 LNB = 05 * inhouse
	DSCC-14	Embedded systems	3	Embedded systems lab	1	
	DSCC-15	Software Engineering	3	Software Engineering Lab	1	
VII		**Internship (Twelve weeks)**	**Credit - 16**			
	DSCC-16	Cryptography and Cyber Security	3	Cryptography and Cyber Security Lab using Python	1	Full Marks = 25 Experiment =15 Viva-Voce = 05 LNB = 05 * inhouse
VIII		Project	16	Project presentation/Viva	4	

Computer and other hardware recommended for laboratory (Upgrade/New installation)

1. **Minimum System requirement**

Computer Hardware upgradation recommended

- **Processor:** Ryzen-3 (3200) series or Ryzen-5 (4600G/5600G) or higher series with compatible motherboard.
  Or
- **Processor:** Intel i-3 10^{th} generation and above, i5 12^{th} generation and above with compatible mother board with integrated graphics.
- **Memory:** DDR-4/5 (3200), 8 GB (minimum recommended) or more
- **Operating System:** Window-10/11 (64 - bit), or Linux (Ubuntu latest version).
- **Open Office/licensed office.**
- Upgrade hard disk to SSD (recommended).

2. **Hardware laboratory**

Digital Circuit lab

- +5V dc Regulated power supply
- Digital multimeter
- Integrated Circuits – 7400, 7402, 7404, 7408, 7410, 7411, 7420, 7432, 7442, 7447, 7446, 7474, 7476, 7483/74283, 7486, 7489/74189, 7490, 74112, 74138, 74147, 74151, 74153, 74157, 74194, 74244, 74373.
- LED.
- Resistors: 100 Ω, 220 Ω, 330 Ω, 470 Ω, 560 Ω, 1K Ω, 1.5K Ω, 2.2K Ω, 4.7K Ω, 10K Ω, 15K Ω, 22k Ω, 100K Ω.
- Semiconductor devices: 1N4007.
- Jumper wires
- Cutters.
- Wish-board or bread board

Semester - I				
Paper	Paper type	Paper name	Credit	Contact hours
DSCC-1	Theory	Computer fundamentals and Digital Logic	3	45
	Practical	Computer fundamentals and Digital Logic lab	1	30
SEC – 1	Theory	Data visualization using spreadsheet	2	30
	Practical	Data visualization using spreadsheet Lab	2	45

CMAM:Theory: (DSCC-1**) Computer Fundamentals and Digital Logic**
Core Course, Theory, Semester – 1, Credits - 03, Contact hours - 45.

Course description:

The course introduces the fundamental principles and concepts of digital logic, which form the foundation of digital systems and computer architecture. Students will learn about Boolean algebra, logic gates, combinational and sequential circuits, and the design and analysis of digital systems.

Course Objectives:

By the end of the course, students should be able to:

1. Understanding of Computer fundamentals, generations, classification of computers and brief understanding of languages used.
2. Understand the principles and terminology of digital logic.
3. Analyze and simplify Boolean expressions using Boolean algebra.
4. Design and implement combinational logic circuits using logic gates.
5. Design and analyze sequential logic circuits, including flip-flops and registers.
6. Apply digital logic concepts to solve practical problems.
7. Utilizing discrete logic gates and integrated circuits on breadboards for the design of digital circuits to enhance hands-on experience and practical understanding.

Computer Fundamentals	
Central Processing Unit (CPU), Primary memory and Secondary Storage devices, I/O devices, generation and classification of Computers: Super, Mainframe, Mini and Personal Computer, System and Application Software, basic concepts on machine, assembly and high-level language.	2 hours
Number Systems	
Weighted and Non - Weighted Codes, Positional, Binary, Octal, Hexadecimal, Binary Coded Decimal (BCD), Gray Codes, Alphanumeric codes, ASCII, EBCDIC, Conversion of bases, signed arithmetic, 1's, 2's complement representation, Parity bits. **Single bit error detection and correcting codes:** Hamming Code. **Fixed- and floating-point Arithmetic**.	3 hours
Boolean Algebra	
Fundamentals of Boolean Expression: Definition of Switching Algebra, Basic properties of Switching Algebra, Huntington's Postulates, Basic logic gates (AND, OR, NOT), De-Morgan's Theorem, Universal Logic gates (NAND & NOR), XOR and others, Minterm, Maxterm, Minimization of Boolean Functions using Karnaugh-Map up to four (4) variables, two level and multilevel implementation using logic gates, simplification of	4 hours

logic expressions.	
Combinational Circuits	
Adder & Subtractor Half adders (2-bit), half Subtractor (2-bit), Full Adder (3-bit), Full Subtractor (3-bit) realization using logic gates, Carry Look Ahead adders, BCD adder, 1's and 2's complement adders/subtractor unit using 4-bit parallel adders.	5 hours
Data Selector/Multiplexer Realization of multiplexers (4 to 1 and 8 to 1) using logical gates, expansion (Cascading), realization of AND, OR and NOT using multiplexers, realization of different Boolean expressions (SOP) using multiplexers.	5 hours
Data Distributor De-multiplexer, Cascading, realization of various functions.	2 hours
Encoders Realization of simple and priority encoders using basic and universal logic gates.	2 hours
Chip Selector/Minterm Generator Realization of decoders using logic gates, function realization, BCD Decoders, Seven Segment display and decoders, cascading.	3 hours
Parity bit, Code Converters and magnitude comparators Parity bit generator/checker, Gray to binary code, binary to Gray code and Gray to Excess-3 code converter, 2 & 3 bit magnitude comparators.	2 hours
Sequential Circuits	
Latch & Flip-Flops Basic Set/Reset (SR) Latch using NAND and NOR gates, Gated S-R latches, Gated D Latch, Gated J-K Latch, race around condition, Master-Slave J-K flip flop, negative and positive clock edge detector circuits, edge triggered SR, D, JK, and T flip flop, flip-flop Conversions.	5 hours
Registers Serial Input Serial Output (SISO), Serial Input Parallel Output (SIPO), Parallel input Serial Output (PISO), Parallel Input Parallel Output (PIPO), Universal Shift Registers.	3 hours
Counters **Asynchronous Counter** UP/DOWN Counters, Mod - N Counters, BCD Counter (Counter Construction using J-K and T Flip Flops).	4 hours
Synchronous Counter UP/DOWN Counters, Mod-N Counters, Ring & Johnson Counters.	3 hours
Integrated Circuits (Qualitative Study): DTL, **TTL:**Concepts of Fan in & out, TTL NOT, TTL NAND & NOR, NMOS, PMOS, CMOS, **IC fabrication (Concepts only):** SSI, MSI, LSI, VLSI, ULSI.	2 hours

CMAM:Practical: (DSCC-1P**) Computer Fundamentals and Digital Logic Lab, Semester – 1, Credits - 01, Contact hours - 30.**

Combinational Circuits

1. Study and prove De-Morgan's Theorem.
2. Realization of Universal functions using NAND and NOR gates.
3. Implementation different functions (SOP, POS) using digital logic gates.

4. Implementation of half (2-bit) and full adder (3-bit) using basic (AND, OR and NOT) and Universal logic gates (NAND & NOR).
5. Design 4 to 1 multiplexer using basic or Universal logic gates and implement half and full adder/subtractor.
6. Design and implement half and full adder/subtractor and other functions using multiplexers 74151/74153 and other necessary logic gates.
7. Cascading of Multiplexers.
8. Design 2 to 4 decoder using basic or universal logic gates, study 74138 or 74139 and implement half and full Adder/Subtractor and other functions.
9. Design a display unit using Common anode or cathode seven segment display and decoders (7446/7447/7448)
10. Design and implement 4-input 3-output (one output as valid input indicator) priority encoder using basic (AND, OR & NOT) logic gates.
11. Design a parity generator and checker using basic logic gates.

Sequential Circuits

1. Realization of SR, D, JK Clocked/Gated, Level Triggered flip-flop using logic gates.
2. Master Slave flip-flop using discrete digital logic gates.
3. Conversion of flip-flops: D to JK, JK to D, JK to T, SR to JK, SR to D Flip-flop.
4. Design asynchronous counters MOD-n (upto 4 bits) UP/ DOWN.
5. Construction Synchronous UP/Down Counter (maximum 4 bits).

Note: The assignments listed below are illustrative examples and not an exhaustive list. They serve as a starting point to cover various aspects of the course.

Recommended Books

1. Digital Fundamentals, 11th Edition by Pearson Eleventh Edition, Thomas L. Floyd.
2. Digital Logic and Computer Design, M Morris Mano, Pearson.
3. Digital Electronics, Principles, Devices and Applications, Anil K. Maini, John Wiley & sons.
4. Digital Principles and Applications, Leach, Malvino, Saha, Tata McGraw Hill Education.
5. Digital Systems, Principal and Applications, Widmer, Moss and Tocci, Pearson.

CMAM: Computer Application - Theory: Data visualization using spreadsheet
SEC-1, Theory, Semester – 1, Credits - 02, Contact hours - 30.

Course Description

This Skill Enhancement Course (SEC) provides a comprehensive introduction to essential concepts and practical skills required for proficient utilization of spreadsheets. Students will gain proficiency in data management, visualization, analysis, and presentation using a widely-used open-source spreadsheet software application such as Open Office, Libre Office, or Google Spreadsheets. Through this course, students will acquire the ability to proficiently create, format, manipulate, and analyze data within spreadsheets to meet a diverse range of needs.

Course Objectives

1. The purpose and potential applications of spreadsheets.
2. Create, format, and modify spreadsheets.

3. Use of formulas, functions, and calculations to perform data visualization.
4. Understanding and utilization of advanced spreadsheet features such as data validation, conditional formatting, and pivot tables.
5. Design visually appealing charts and graphs to represent data.
6. Collaborate and share spreadsheets with others.
7. Apply spreadsheet skills to real-world scenarios and problem-solving.
8. Role of spreadsheets in data analysis.
9. Import, clean, and transform data for analysis.
10. Applicability of statistical and mathematical functions for data visualization.
11. Advanced features and tools for data visualization.
12. Perform exploratory data analysis and identify patterns and trends.
13. Create informative reports and summaries based on data analysis.
14. Apply data analysis techniques to real-world problems.

Description	Teaching hours
Introduction to Spreadsheets Spreadsheets and their applications, overview of spreadsheet software (e.g., Open office, Google Sheets, Excel), creating workbooks, modifying workbook, modifying workbook, zooming in on a worksheet, arranging multiple workbook windows, adding buttons to the quick access toolbar, customizing the ribbon, maximizing usable space in the program windownavigating the spreadsheet interface, entering and editing data in cells saving, opening, and closing spreadsheet files.	2 hours
Working with Data and Tables Entering and revising data, moving data within a workbook, finding and replacing data, correcting and expanding upon worksheet data, defining tables.	2 hours
Performing Calculations on Data Naming groups of data, creating formulas to calculate values(e.g., SUM, AVERAGE, COUNT), summarizing data that meets specific conditions (e.g., AVERAGEIF, COUNTA, COUNTBLANK, COUNTIFS, SUMIF, IFERROREtc), finding and correcting errors in calculations.	2 hours
Changing Workbook Appearance Formatting Cells, defining styles, workbook themes and table styles, making numbers easier to read, changing the appearance of data based on its value, adding images to worksheets.	2 hours
Data Analysis and Manipulation Limiting data appearance on screen, working with text functions for data cleaning, Splitting and combining data, Data normalization and standardization, working with ranges and named ranges, conditional formatting, data validation and error checking, using logical functions (e.g., IF, AND, OR), sorting and filtering data.	2 hours
Advanced Spreadsheet Features Creating and managing tables, creating and modifying pivot tables, using lookup functions (e.g., VLOOKUP, HLOOKUP), working with charts and graphs, importing and exporting data.	2 hours
Statistical Functions and Analysis Descriptive statistics (mean, median, mode, variance, etc.), Calculating measures of central tendency and dispersion, Correlation and regression analysis, Hypothesis testing and confidence intervals, Analysis of variance (ANOVA).	2 hours
Pivot Tables and Data Aggregation	

Creating pivot tables for data summarization, grouping and aggregating data by categories, applying filters and slicers to pivot tables, calculating calculated fields and items.	3 hours
Advanced Data Visualization Creating charts and graphs for data representation, customizing chart elements (titles, axes, legends), Using sparklines and data bars for visual analysis, creating interactive dashboards, incorporating trendlines and forecasting in charts.	3 hours
Exploratory Data Analysis Identifying patterns and outliers in data, creating histograms and box plots, using conditional formatting for data visualization, Data segmentation and drill-down analysis, Applying data validation rules for data integrity.	3 hours
Advanced Analysis Techniques Using goal seek and solver for optimization problems, performing "what-if" analysis with data tables, simulating data using random number functions, Monte Carlo simulation for risk analysis, creating scenario analysis models.	3 hours
Reporting and Presentation of Results Designing informative reports and summaries, creating interactive dashboards for data presentation, data visualization best practices, documenting data analysis processespresenting findings to stakeholders.	2 hours
Collaboration and Sharing Protecting worksheets and workbooks, sharing spreadsheets with others, tracking changes and commenting, collaborating in real-time, using version history and revision control.	2 hours

CMAM: Practical - Data visualization using spreadsheet
SEC-1P, Laboratory, Semester – 1, Credits - 02, Contact hours - 45.

1. Create a personal budget spreadsheet that tracks income, expenses, and savings over a specified period. Use formulas and functions to calculate totals, percentages, and remaining balances.

2. A dataset containing sales data for a company to be provided. A spreadsheet to be created that calculates monthly sales totals, identifies top-selling products, and visualizes sales trends using line charts or bar graphs. Use conditional formatting to highlight exceptional sales performances.

3. Design a grade book spreadsheet that calculates students' final grades based on assignments, exams, and participation. Incorporate weighted grading systems, formulas for calculating averages, and conditional formatting to indicate performance levels. Generate reports to track individual student progress.

4. Create a spreadsheet that tracks inventory for a hypothetical business. Include columns for item names, quantities, prices, and total values. Use formulas to automatically update inventory totals, generate alerts for low stock, and create visualizations to represent inventory levels over time.

5. Loan parameters, such as principal amount, interest rate, and loan term to be provided. Create a spreadsheet that calculates monthly loan payments, remaining balances, and interest paid over time using appropriate formulas. Create a chart to visualize the loan's repayment schedule.

6. Dataset to be provided which will allow various data analysis tasks using spreadsheets. Calculation of summary statistics, sorting and filtering data, creating pivot tables for deeper insights, and generation of charts or graphs to visualize patterns or trends within the data.

7. A dataset to be selected (e.g., stock prices, weather data, population growth, etc) and create line charts or area charts to visualize trends over time. Students should choose appropriate chart types, label axes, and add titles and legends to make the visualization clear and informative.

8. A dataset containing information about different products or variables (e.g., sales data, customer satisfaction ratings) to be provided and following to be done; create bar charts or column charts to compare the performance or rankings of the items. Use color, data labels, and chart elements to enhance the visual comparison.

9. A dataset containing time-series data for multiple variables (e.g., monthly sales data for different products) to be provided and the following task to be performed;to create a combo chart with lines and columns to compare the trends of the variables and identify any relationships or patterns.

10. To create a unique visualization using advanced spreadsheet features and tools. For example, an experiment with sparklines, radar charts, or treemaps to represent specific types of data or explore innovative ways to visualize information.

Note: The assignments listed below are illustrative examples and not an exhaustive list. They serve as a starting point to cover various aspects of the course.

Recommended Text books

1. Data Analysis and Decision Making with Microsoft Excel" by S. Christian Albright.
2. Microsoft Excel 2019Data Analysis andBusiness Modeling, Sixth Edition,Wayne L. Winston, Pearson education.
3. Excel 2019 Bible, Michael Alexander, 11th edition, Wiley.
4. Microsoft Office 2019 for Dummies, Wallace Wang, Wiley.

Recommended Application Software

1. Google Spreadsheets
2. Libre/Open Office
3. Excel sptreadsheets

Semester - II				
Paper	**Paper type**	**Paper name**	**Credit**	**Contact hours**
DSCC-2	Theory	Problem Solving using C	3	45
	Practical	Problem Solving using C Lab	1	30
SEC – 2	Theory	Web Development	2	30
	Practical	Web Development Lab	2	45

CMAM: Computer Application - Theory: Problem Solving using C
DSCC-2, Theory, Semester – 2, Credits - 03, Contact hours - 45.

Objective of the Course

The objectives of this course are to make the student understand programming language, programming, concepts of Loops, reading a set of Data, stepwise refinement, Functions, Control structure, Arrays. After completion of this course the student is expected to analyze the real-life problem and write a program in 'C' language to solve the problem. The main emphasis of the course will be on problem solving aspect i.e. developing proper algorithms.

After completion of the course the student will be able to;

1. Develop efficient algorithms for solving a problem.
2. Use the various constructs of a programming language viz. conditional, iteration and recursion.
3. Implement the algorithms in "C" language.
4. Use simple data structures like arrays, stacks and linked list in solving problems.
5. Handling File in "C".

Outline of Course

S. No.	Topic	Minimum number of hours
1	Introduction to Programming	03
2	Algorithm/ Flowchart for Problem Solving	06
3	Introduction to 'C' Language	02
4	Conditional Statements and Loops	05
5	Arrays	05
6	Functions	06
7	Storage Classes	02
8	Structures and Unions	05
9	Pointers	06
10	File Processing	03
11	Organizing C Projects	02
Lectures = 45 **Practical/tutorials = 30, Total = 75**		

Detailed Syllabus

Description	Teaching hours
Introduction to Programming The Basic Model of Computation, Algorithms, Flow-charts, Programming Languages, Compiler, Interpreter, Assembler, Linker and Loader, Testing and Debugging, Documentation.	03 hours
Algorithms/ Flowchart for Problem Solving Exchanging values of two variables, summation of a set of numbers, decimal base to binary base conversion, reversing digits of an integer, GCD (Greatest Common Division) oftwo numbers, test whether a number is prime, organize numbers in ascending order using bubble sort, find integer square root of a number, factorial computation, Fibonacci sequence, evaluate 'sin x' as sum of a series, reverse order of elements of an array, find largest number in an array, print elements of upper triangular matrix, multiplication of two matrices, evaluate a Polynomial.	06 hours
Introduction to 'C' Language Character set, variables, identifiers and their nomenclature, built-in data types, variable declaration, arithmetic operators and expressions, constants and literals, simple assignment statement, basic input/output statement, simple 'C' programs.	02 hours
Conditional Statements and Loops Decision making within a program, conditions, relational operators, logical connectives, if statement, if-else statement, **Loops:** while loop, do while, for loop, nested structure, infinite loops, switch-case, break, continue statement, structured programming.	05 hours
Arrays One dimensional array: Array manipulation; Searching, Insertion, deletion of an element from an array; finding the largest/smallest element in an array; two dimensional arrays, addition/multiplication of two matrices, Transpose of a square matrix; null terminated strings as array of characters, standard library string functions.	05 hours
Functions Top-down approach of problem solving, modular programming and functions, standard library of C functions, Prototype of a function: Formal parameter list, return type, function call, block structure, passing arguments to a function: call by reference, call by value, Recursive functions, arrays as function arguments.	06 hours
Storage Classes Scope and extent, Storage Classes in a single source file: auto, extern and static, register, Storage Classes in multiple source files: extern and static	02 hours
Structures and Unions Structure variables, initialization, structure assignment, nested structure, structures and functions, structures and arrays: arrays of structures, structures containing arrays, unions.	05 hours
Pointers Address operators, pointer type declaration, pointer assignment, pointer initialization, pointer arithmetic, functions and pointers, Array of Pointers, pointer to an array, pointers and structures, dynamic memory allocation.	06 hours
File Processing Concept of Files, File opening in various modes and closing of a file, reading from a	03 hours

file, writing onto a file, appending to a file.	
Organizing C projects, working with multiple source directories, makefiles.	02 hours

Recommended books main reading

1. Byron S Gottfried “Programming with C” Second edition, Tata McGraw Hill, 2007 (Paperback)
2. R.G. Dromey, “How to solve it by Computer”, Pearson Education, 2008.
3. Kanetkar Y, “Let us C”, BPB Publications, 2007.
4. Hanly J R & Koffman E.B, “Problem Solving and Program design in C”, Pearson Education, 2009.
5. Kashi Nath Dey and Samir Bandyopadhayay “C Programming Essentials” Pearson India Education, 2010.

Supplementary reading.

1. E. Balagurusamy, “Programming with ANSI-C”, Fourth Edition,2008, Tata McGraw Hill.
2. Venugopal K. R and Prasad S. R, “Mastering ‘C’”, Third Edition, 2008, Tata McGraw Hill.
3. B.W. Kernighan & D. M. Ritchie, “The C Programming Language”, Second Edition, 2001, Pearson education.
4. ISRD Group, “Programming and Problem-Solving Using C”, Tata McGraw Hill,2008.
5. Pradip Dey, Manas Ghosh, “Programming in C”, Oxford University Press, 2007.

CMAM: Computer Application - Practical: Problem Solving using C
DSCC-2P, Practical, Semester – 2, Credits - 01, Contact hours - 30.

Algorithms / Flowchart (Sample and simple assignments)

1. Design a flowchart/ Algorithm for a basic calculator that accepts two numbers and an operator (+, -, *, /) as input from the user and performs the corresponding operations, and displaying/print the result.
2. Create a flowchart/Algorithm that converts a temperature from Celsius to Fahrenheit or vice versa based on user input.
3. Design a flowchart/Algorithm that calculates the factorial of a given positive integer provided by the user.
4. Create a flowchart/Algorithm that finds and displays the largest number among three input numbers given by the user.
5. Design a flowchart/Algorithm to implement the linear search algorithm to find a specific element in an array of integers. The array and the element to search for should be taken as user input.
6. Create a flowchart/Algorithm that calculates the area and perimeter/circumference of different shapes (e.g., circle, rectangle, triangle) based on user input for dimensions.
7. Design a flowchart/Algorithm that checks whether a given input string is a palindrome or not.

Introduction to ‘C’ Language (Assignments/examples related to simple C program.)

8. Write a program in C to read two numbers and produce the sum and product of those numbers and show the result separately.
9. Write a program in C to read two numbers and print the greater number, if both the numbers are same them print “EQUAL”.
10. Write a program in C multiple numbers say n and print the greatest and the third greatest.
11. Write a program in C to read n numbers and print the even/odd numbers up to n.
12. Write a program in C to read a number and print the sum of n natural numbers.
13. Write a program in C to read a number n and print factor of n.

14. Write a program in C to read a number n and print first 10 multiples of n.
15. Write a program in C to read a number n and print if n is “PRIME” or “COMPOSITE”.
16. Write a program in C to calculate the average of a set of N numbers.
17. Write a program in C convert the temperature given in Celsius to Fahrenheit or vice-versa.
18. Write a program in C to determine and print the sum of the following harmonic series for a given value of n: 1+1/2+1/3+……..1/n.
19. Write a program in C that reads a floating-point number and then displays the right most digits of integral part of the number.
20. Write a program in C to accept the length and breadth in meters and calculate the area and perimeter and also determine if it is a rectangle or a square based on the inputs given.
21. Write a program in C to accept an input and determine if the input entered is a number or alphabet or a special character.
22. Write a program in C to accept a word and then print the reverse case that is lower to upper or upper to lower case.
23. Write an interactive program in C which will demonstrate the process of division/multiplication, the user should be asked to enter two-digit numbers.

Conditional Statements and Loops (simple examples)

24. Write a program in C to read a number n and print n terms of the Fibonacci series.
25. Write a program in C to read a number n and print a single digit answer showing sum of the digits of n. (example – input 8626, expected output – 4, explanation 8+6+2+6 = 22, 2+2 = 4).
26. Write a program in C to read a number n and print all the prime numbers up to n.
27. Write a program in C to read a number n and print the following pattern (input = 5, expected output
    1
    12
    123
    1234
    12345).
28. Write a program in C to check if the given number is the Armstrong number or not (e.g 153 = 13+53+33).
29. Write a program in C to check the type of the given triangle whether it is equilateral, isosceles or scalene.

Arrays (examples of few simple programs)

30. Write a program in C to read a string and store it into a character array. Check whether the string is a palindrome or not and display accordingly.
31. Write a program in C to read a list of numbers stored in an integer array and while saving them arrange in ascending order.
32. Write a program in C to read two matrices and perform addition.
33. Write a program in C to read two matrix and check their compatibility for multiplication, if compatible then find product and print it.
34. Write a program in C to read a string and print the triangular pattern using the string.

Functions

35. Write a program in C to print all the Armstrong number from 1 to 500.
36. Write a function ***convert ()*** that returns a weight in Kg after being given a weight in pounds.
37. Write a function to find all perfect numbers from 1 to 100 (perfect numbers are positive integers where the sum of perfect divisor is the number itself, e.g., 6 = 1+2+3).
38. Write a function power () to find base raise to power [$\mathbf{base^{power}}$].

39. Write a program in C to find solution of a quadratic equation [$x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$] where values a, b and c to be accepted from the user as input.
40. Accept inputs from the user and echo it on to the screen in normal as well as in reverse using void recursive function.
41. Accept any number from the user and calculate the factorial of the number using recursion
42. Accept numbers n and print the odd/even numbers up to n using recursive function.
43. Write a program in C in compute the cubes of all numbers from 10 to 20.
44. Write a program in C to find the GCD of a number.
45. Write a program in C to generate all combinations of 1, 2, 3, 4 using recursion, e.g.,1234, 2341… etc.

Storage Classes

46. Write a program in C to accept a number and find the factorial of the number demonstrating use of automatic variables.
47. Write a program in C to accept two numbers and find the sum of the number demonstrating use of external variables.
48. Write a program in C to accept two numbers and find the sum of the number demonstrating use of global variables.
49. Write a program in C to illustrate the use of static variables.
50. Write a program in C to accept numbers till a negative number is entered and calculate the sum of a list of numbers read using static variable.
51. Write a program in C to sum integers and use static variable to store the cumulative sum.

Pointers

52. Write a program in C to swap two numbers of n length.
53. Write a program in C for swapping numbers using functions.
54. Write a program in C to illustrate the Call by Value and Call by reference a rule in C programming.
55. Write a program in C to use a double dimensional array and print each cells value and address.
56. Write a program in C to show the use of Array, declared at compilation time (static manner) to read 10 numbers and display them.
57. Write a program in C to show the use of Array, declared dynamically to read 10 numbers and display them.
58. Write a program in C to read a string in a dynamic array and determine whether it is palindrome or not.

Structures and Unions

59. Write a program in C to read the data of a student, store it in a structure and display it.
60. Write a program in C to read the data of many students, store it in a structure and display the student's data and average percentage of the class.
61. Write a program in C to accept two dates from the user, validate both of them and check if they are different dates.
62. Write a program in C to accept students' data from the user. Check if the student stream is science, commerce or arts. If the stream is arts, then print the class of students. If the stream is science, then print the grade and if the stream is commerce, then print the percentage.

Files

63. Write a program in C showing the technique of opening and closing a file say **result.dat** and writing a list of numbers and its square into the file.

64. Write some texts into a file, reopen the file in read mode and reproduce the text on the monitor (use of putc() and fputc()).
65. Write a few numbers in the file created earlier. Reopen it in Read mode, write odd numbers in one file and even number in another file (use the **getw** and **putw** functions).
66. Write programs to demonstrate the use of getc(), fgetc() and ungetc().
67. Write programs to demonstrate the use of String I/O, Formatted I/O and End of file eof() and feof().

Recommended assignment content/structure

- Objective
- Algorithm/Flowchart
- Code
- Result
- Conclusion

Platform/Compiler

- GCC

Note: The assignments listed below are illustrative examples and not an exhaustive list. They serve as a starting point to cover various aspects of the course.

CMAM: Theory: Web development
SEC-2, Theory, Semester – 2, Credits - 02, Contact hours - 30.

Course Description

This course provides an introduction to web development using HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets). Students will learn the core concepts and practical skills needed to create and style web pages. The course covers the fundamentals of HTML structure, CSS styling properties, and responsive web design principles.

Course Objectives

1. Understanding the basics of web development and the role of HTML and CSS.
2. Create well-structured HTML documents using proper tags and elements.
3. Apply CSS to style web pages, including layout, typography, colors, and images.
4. Implement responsive design techniques to ensure optimal display on different devices.
5. Incorporate multimedia elements, such as images, videos, and audio, into web pages.
6. Understand best practices for organizing and maintaining code in web development projects.
7. Develop and deploy a basic website using HTML and CSS.

Description	Teaching hours
Introduction to Web development Overview of web technologies and the role of HTML and CSS, understanding the structure of a web page, introduction to web browsers and developer tools.	2 hours
HTML Fundamentals	

Introduction to HTML tags and elements, creating headings, paragraphs, lists, and links, working with images and multimedia content, creating forms for user input.	2 hours
CSS basics Introduction to CSS and its role in web page styling, selectors, properties, and values, applying inline, internal, and external style sheets, formatting text, backgrounds, and borders.	2 hours
CSS Layout and box model Understanding the box model and its impact on layout, working with margins, padding, and borders, positioning elements using floats, positioning properties, and flexbox, creating responsive layouts with media queries.	2 hours
Typography and colors Styling text with fonts, sizes, weights, and styles, formatting text using CSS properties, understanding color models and applying colors to elements.	3 hours
Images and multimedia Working with images: sizing, aligning, and optimizing, incorporating videos and audio into web pages, implementing responsive images and media.	3 hours
CSS Selectors and specificity Understanding CSS selectors and specificity, applying styles to specific elements and classes, using pseudo-classes and pseudo-elements.	3 hours
Responsive Web design Introduction to responsive design principles, creating fluid layouts using CSS media queries, adapting web pages for different screen sizes and devices.	3 hours
CSS Frameworks and libraries Overview of popular CSS frameworks (e.g., Bootstrap, Foundation), using pre-built CSS components and grids, customizing and integrating CSS frameworks into web projects.	2 hours
Web development best practices Organizing and structuring code files and directories, validating HTML and CSS code, optimizing web pages for performance, introduction to version control with Git.	2 hours
Building and deploying a website Planning and designing a basic website structure, Implementing HTML and CSS to create the website, testing and debugging the website across different browsers, deploying the website to a local host/web server.	6 hours

CMAM: Web development Lab
SEC-2P, Laboratory, Semester – 2, Credits - 02, Contact hours - 45.

1. Creating a personal portfolio website using HTML and CSS. There should be sections for an about me, projects, skills, and contact information's. Using CSS to style the layout, typography, and colors to create a visually appealing and professional-looking portfolio.

2. To design a responsive website that adapts to different screen sizes. They should create a layout that adjusts fluidly using CSS media queries and responsive design techniques.

3. To create a product landing page for a fictional product or an existing one. HTML to be used to structure the page and CSS to style the layout, typography, buttons, and images. Main focus to be on creating an engaging page that effectively showcases the chosen product.

4. To incorporate CSS animation effects into a web page. Use CSS transitions, transforms, and keyframe animations to add interactive and engaging elements to the website. Create animations for hover effects, scrolling effects, image sliders, or menu transitions.

5. Redesign an existing website using HTML and CSS. Analyze the original design and propose improvements to the layout, typography, color scheme, and overall user experience.

6. Create a webpage layout using CSS Flexbox or CSS Grid. Design a responsive layout that organizes content in a visually appealing way. Experiment can be performed with different grid or flexbox properties to create flexible and responsive designs.

7. To design and style an interactive form using HTML and CSS. They should incorporate various form elements such as text inputs, checkboxes, radio buttons, and select dropdowns. Apply CSS styling to improve the form's visual appearance and user experience.

Note: The assignments listed below are illustrative examples and not an exhaustive list. They serve as a starting point to cover various aspects of the course.

Suggested Readings.

1. Mastering HTML, CSS & Java Script Web Publishing, Laura Lemay, Rafe Colburn, Jennifer Kyrnin, BPB Publication.
2. Web designing and development, Satish Jain, BPB Publications.
3. HTML & CSS: Thecompletereference, Thomas Powell, McGraw Hill education.
4. Web programming with HTML5, CSS and JavaScript, John Dean, Joneas and Bartlet learning.
5. SamsTeachYourself HTML, CSS, and JavaScript AllinOne, Julie C Meloni, Pearson Education.
6. Learning Web App development, Semmy Purewal, O'Reilly.

Semester - III				
Paper	**Paper type**	**Paper name**	**Credit**	**Contact hours**
DSCC-3	Theory	Data Structure	3	45
	Practical	Data Structure using C Lab	1	30
DSCC-4	Theory	Programing in Python	3	45
	Practical	Python lab	1	30
SEC – 3	Theory	Mobile App Development	2	30
	Practical	Mobile App Development Lab	2	45

CMAM: Theory: Data Structure
DSCC-3, Theory, Semester – 3, Credits - 03, Contact hours - 45.

Course Objectives:

The objective of the Data Structures course is to introduce students to fundamental concepts and practical applications of various data structures, enabling them to analyze and implement efficient solutions for complex problems. The course aims to develop students' understanding of arrays, linked lists, stacks, queues, trees, graphs, and hash tables, and to equip them with the skills to evaluate and select appropriate data structures based on performance metrics.

Course Outcomes

By the end of the course, students will be;

- Proficient in designing,
- Coding,
- optimizing algorithms,
- demonstrating readiness for advanced studies and real-world applications in computer science.
- They will also enhance their problem-solving abilities, collaborate effectively, and communicate technical information clearly.

Description	Teaching hours
Introduction to Data Structure Abstract Data Type.	**01 hour**
Arrays 1D, 2D and Multi-dimensional Arrays, Sparse Matrices. Polynomial representation.	**05 hours**
Linked Lists Singly, Circular and Doubly Lists, Polynomial representation.	**03 hours**
Stacks Array and linked representation of stack, Prefix, Infix and Postfix expressions, utility and conversion of these expressions from one to another, evaluation of postfix and prefix expression using stack, applications of stack, limitations of Array representation of stack.	**09 hours**
Queues Array and Linked representation of Queue, Circular Queue, De-queue, Priority Queues.	**05 hours**
Images and multimedia Working with images: sizing, aligning, and optimizing, incorporating videos and audio	**05 hours**

into web pages, implementing responsive images and media.	
Developing Recursive Definition of Simple Problems and their implementation; Advantages and Limitations of Recursion; Understanding what goes behind Recursion (Internal Stack Implementation), Tail recursion.	**05 hours**
Trees Introduction to Tree as a data structure: Binary Trees (Recursive and Iterative Traversals), Binary Search Tree (Traversal, Insertion, Deletion and Searching), Threaded Binary Trees(Traversal and advantages).	**15 hours**
Searching and Sorting Linear Search, Binary Search, Comparison of Linear and Binary Search with respect to timecomplexity, Selection Sort, Bubble sort, Insertion Sort, Merge Sort, Quick sort, Heap sort,Shell Sort, Radix sort, Comparison of Sorting Techniques with respect to time complexity.	**10 hours**
Hashing Introduction to Hashing, Different hashing Techniques, Collision and resolving collision by Open Addressing, Closed Hashing, Separate Chaining, Choosing a Hash Function.	**05 hours**

CMAM: Computer Application - Practical: Data Structure using C
DSCC-3P, Practical, Semester – 3, Credits - 01, Contact hours - 30.

1. **Array Operations**
    - Implement basic operations on arrays: insertion, deletion, and searching.
    - Write a program to find the maximum and minimum elements in an array.
    - Implement sorting algorithms (e.g., bubble sort, insertion sort).
2. **Linked Lists**
    - Create a singly linked list and perform operations like insertion, deletion, and traversal.
    - Implement a function to reverse a linked list.
    - Create a doubly linked list and implement similar operations.
3. **Stacks**
    - Implement a stack using arrays and linked lists.
    - Write functions for stack operations: push, pop, and peek.
    - Use stacks to evaluate postfix expressions.
4. **Queues**
    - Implement a queue using arrays and linked lists.
    - Write functions for queue operations: enqueue, dequeue, and peek.
    - Implement a circular queue and a priority queue.
5. **Trees**
    - Implement a binary tree with operations like insertion, traversal (in-order, pre-order, post-order), and deletion.
    - Write a program to find the height of a binary tree and check if it is balanced.
    - Implement a binary search tree (BST) and functions for insertion, search, and deletion.
    -

6. **Graphs**
    - Implement a graph using adjacency matrix and adjacency list representations.
    - Write programs for graph traversal algorithms: depth-first search (DFS) and breadth-first search (BFS).
    - Implement Dijkstra's algorithm for finding the shortest path in a weighted graph.
7. **Hash Tables**
    - Implement a hash table with basic operations: insertion, deletion, and searching.
    - Handle collisions using chaining or open addressing.
8. **Heaps**
    - Implement a binary heap and operations like insertion, deletion, and heapify.
    - Write a program to perform heap sort using the heap data structure.
9. **Graphs Algorithms**
    - Implement Kruskal's or Prim's algorithm for finding the Minimum Spanning Tree (MST).
    - Write a program to detect cycles in a graph.

Each of these assignments can be adjusted in complexity depending on the skill level of the students.

CMAM: Programming in Python
DSCC-4, Theory, Semester – 3, Credits - 03, Contact hours - 45.

Course Objective for Programming in Python

The **Programming in Python** course aims to provide students with a comprehensive foundation in Python programming, enabling them to create efficient and effective software solutions.

By the end of the course:

1. students will have a thorough understanding of Python's syntax, semantics, and core programming constructs.
2. They will develop problem-solving skills by applying logical thinking and designing algorithms in Python.
3. Students will learn to use control structures like loops and conditional statements to manage program flow, write modular and reusable code using functions and modules, and handle files and exceptions to build robust programs.
4. The course also covers object-oriented programming principles, including the creation of classes and objects, inheritance, and polymorphism.
5. Students will explore advanced topics such as regular expressions, web scraping, and data visualization.

Description	Contact hours
Introduction History of Python Programming Language, Installing Anaconda Python distribution, Installing and using Jupyter Notebook, features of Python, built in Object Types, libraries and tools, **Paradigms:** Procedural, Object-Oriented, Functional.	**01 hour**

Python programming and its parts Identifiers, Keywords, Statements and Expressions. **Variables:** legal variable names, assigning values to variables. **Operators:** Arithmetic, Assignment, Comparison, Logical, Bitwise, Precedence and Associativity. **Data Types:** Numbers, Boolean, None, Indentation, **Strings: String Operators:** Concatenation Operator (+), Replication Operator, Membership Operator. Functions for String Handling: len(), Capitalize(), find(), count, Endswith(), Encode, Decode, Miscellaneous Functions. **Comments:** Single Line Comment, Multiline Comments, Reading Input, Print Output, str.format() Method, f-strings. **Type Conversions:** int() function, float() function, str() function, chr() function, complex() function, ord() function, hex(), oct() function, type() function and is operator, dynamic and strongly typed language. Lists and Tuples, List, Tuples, Features of Tuples.	**05 hour**
Conditional Statements Introduction, if, if-else, and if-elif-else constructs, if-elif-else Ladder, Logical Operators, Ternary Operator, get Construct. **Looping** Introduction, While, Patterns, Nesting and applications of loops in lists.	**03 hour**
Functions **Features of a function:** modular programming, reusability of code, manageability. **Basic terminology:** Name of a Function, Arguments, Return Value.**Definition and Invocation:** Working. **Types of Function:** Advantage of Arguments, Implementing Search, Scope, **Recursion:** Rabbit Problem, disadvantages of using Recursion.	**05 hour**
Iterations, Generators, and Comprehensions Power of "For", Iterators, Defining an Iterable Object, Generators, Comprehensions.	**02 hours**
File Handling File handling mechanism, Open function and file access modes. **Python Functions for File Handling:** Essential Ones, OS Methods, Miscellaneous Functions and File Attributes.Command Line Arguments.	**04 hours**
Introduction to Object Oriented Paradigm Creating new types, Attributes and Functions, **Elements of Object-Oriented Programming:** Class, Object, Encapsulation, Data hiding, Inheritance, Polymorphism, Reusability. **Classes and Objects** Introduction to Classes, defining a Class, Creating an Object, Scope of Data Members, Nesting, Constructor, Constructor Overloading, Destructors.	**05 hours**
Inheritance Introduction to Inheritance and Composition, Inheritance and Methods, Composition. **Inheritance Importance and Types:** Need for Inheritance, Types of Inheritance. **Methods:** Bound, Unbound, Methods are Callable Objects, Importance and Usage of Super, Calling the Base Class Function Using Super. Search in Inheritance Tree, Class Interface and Abstract Classes.	**05 hours**
Operator Overloading Introduction, **_init_ Revisited**: Overloading _init_ (sort of). Methods for Overloading Binary Operators, **Overloading Binary Operators:** The Fraction Example, Overloading the += Operator, Overloading the > and < Operators, Overloading the _boolean_ Operators: Precedence of _bool_over _len_, Destructors.	**05 hours**
Exception Handling **Importance and Mechanism:** Try/Catch, Manually Raising Exceptions**.** **Built-In Exceptions in Python, The Process:** Exception Handling: Try/Except,	**04 hours**

Raising Exceptions. Crafting User Defined Exceptions.	
Introduction to NUMPY Introduction to NumPy and Creation of a Basic Array, **Functions for Generating Sequence:** arange(), linspace(), logspace(). Aggregate Functions, Broadcasting, Structured Arrays.	**04 hours**
Introduction to MATPLOTLIB Plot Function, Subplots, 3-Dimensional Plotting.	**02 hours**

CMAM: Programming in Python Lab
DSCC-4P, Practical, Semester – 1, Credits - 01, Contact hours - 30.

Sample Programs

1. Write a program to swap two numbers.
2. Ask the user to enter the coordinates of a point and find the distance of the point from the origin.
3. Ask the user to enter two points (x and y coordinates) and find the distance between them.
4. Ask the user to enter three points and find whether they are collinear.
5. In the above question, if the points are not collinear then find the type of triangle formed by them (equilateral, isosceles or scalene).
6. Ask the user to enter two points and find if they are at equal distances from the origin.
7. Ask the user to enter 4 points and arrange them in order of their distances from the origin.
8. Ask the user to enter a number and find the number obtained by reversing the order of the digits.
9. Ask the user to enter a four-digit number and check whether the sum of the first and the last digits is same as the sum of the second and the third digits.
10. Ask the user to enter the concentration of hydrogen ions in a given solution(C) and find the PH of the solution using the following formula.
    $$PH = \log_{10} C$$
    If the PH is <7 then the solution is deemed acidic, else it is deemed as basic.
    Find if the given solution is acidic.In the above question find whether the solution is neutral. (A solution isneutral if the PH is 7).
11. The centripetal force acting on a body (mass m), moving with a velocity v,in a circle of radius r, is given by the formula $\mathbf{mv^2/r}$. The gravitational forceon the body is given by the formula (GmM)/R2, where m and M are themasses of the body and earth and R is the radius of the earth. Ask the user toenter the requisite data and find whether the two forces are equal or not.
12. Ask the user to enter his salary and calculate the TA, DA, which is 10% of thesalary; the HRA, which is 20% of the salary and the gross income, which isthe sum total of the salary, TADA and the HRA.
13. Ask the user to enter a number and find whether it is a prime number.
14. Ask the user to enter a number and find all its factors.
    Example: If number = 30, then factors are 2, 3, and 5.
15. Ask the user to enter two numbers and find the lowest common multiple, example: If numbers are 30 and 20, then LCM is 60, as both 20 and 30 are factors of 60.
16. Ask the user to enter two numbers and find the highest common factor.
17. Write a generator that produces the terms of a geometrical progression.
    also write the corresponding iterator class.
18. Write a generator that produces the terms of a harmonic progression.
    also write the corresponding iterator class.
19. Write a generator that produces all the prime numbers up to a given number.
    also write the corresponding iterator class.

20. Write a generator that produces all the Fibonacci numbers up to n.
21. Write a generator that produces all the Armstrong numbers up to n.
    also write the corresponding iterator class.
22. Write a generator that produces Pythagoras triples in the range (1, 20).
23. Write a generator that produces all the multiples of 6 up to the given number.
24. Write a program to copy the contents of one file to another.
25. Write a program to capitalize the first character of each word in a file.
26. Write a program to find the ASCII value of each character in a file.
27. Write a program to find the frequency of each character in a file.
28. Write a program to find all occurrences of a word, entered by the user, in a given file.
29. Write a program to replace a given character with another in a file.
30. Write a program to replace a given word with another, in a given file.
31. Write a program to find the frequency of a given word in a file.
32. Write a program to find the word used the minimum number of times in a given file.
33. Write a program to change the name of a file to the name entered by the user.
34. Write a program to create a directory and then create a new file in it.
35. Write a program to print the name, number of characters, and number of spaces in a file.
36. Write a program to convert the characters of a given file to binary format.
37. Write a program to find the words starting with a vowel from a given file.
38. Write a program to implement any substitution cipher on the text of a given file.
39. Write a program to find the sum of ASCII values of the characters of a given string.
40. Write a program to find a particular substring in a given string.
41. Write a program to split a given text into tokens.
42. Write a program to check which of the tokens obtained in the above question are keywords.
43. Write a program to convert a string entered by a user to that obtained by adding “k” to each character’s ASCII value.
44. Create a class called distance having meter and centimeter as its data members. The member functions of the class would be putData(), which takes the values of meter and centimeter from the user; putData(), which displays the data members and add, which adds the two distances. The addition of two instances of distances (say d1 and d2) would require addition of corresponding centimeters (d1.centimeter +s2.centimeter), if the sum is less than 100, otherwise it would be (d1.centimeter +s2.centimeter)%100. The “meter” of the sum would be the sum of meters of the two instances (d1.meter +d2.meter), if (d1.centimeter +d2.centimeter) <100, otherwise it would be (d1.meter +d2.meter+1).
45. Overload the + operator for the above class. The + operator should carry out the same task as is done by the add function. The subtraction of two instances of distances (say d1 and d2) would require the subtraction of corresponding centimeters (d1.centimeter -s2.centimeter). The “meter” of the difference would be the sum of meters of the two instances (d1.meter -d2.meter).
46. Overload the – operator for the distance class. Assume that d1-d2, would always mean d1>d2.
47. Ask the user to enter the value of n. Create an array, a, containing integers from 0 to (n-1).
48. Create another array, b, from the above array containing all the even numbers of the original array.
49. Create an array, c, from an array containing all the odd numbers of the original array.
50. Now add b and c and divide each element of the resultant array by 2. Check if the result is same as a.
51. Create a one-dimensional array containing 500 random numbers.
52. Find the mean, standard deviation, median, 25th percentile, 75th percentile of the numbers.
53. Create a histogram of the above data with 10 bins.
54. Implement linear search. Also use the requisite method of NumPy and compare the running time of both.

55. Sort the elements of the array. Also use the requisite method of NumPy and compare the running time of both.
56. Create an array of 500 random numbers. Find the product of the numbers using loops and by using the functions of numpy and compare the running time by the two methods.
57. From the above array find the maximum element by the following methods:
    (a) Using the maximum function of NumPy
    (b) Using loops in O(n) time
    (c) Using divide and conquer in O(log n) time
58. Create a two-dimensional array having n rows and m columns containing:
    (a) All ones
    (b) All zeros
    (c) 1's at the diagonal
    (d) 0-(m-1) at the diagonal
    (e) Random numbers
59. Create a two-dimensional array having 7 rows and 7 columns such that an element aij (element at the ith row and the jth column) is (i+j)2.
60. Create a list having numbers in arithmetic progression. Ask the user to enter the first term, the common difference and the number of terms of the arithmetic progression. Plot the values using the plot function.
61. Create a list having numbers in geometric progression. Ask the user to enter the first term, the common ratio and the number of terms of the geometric progression. Plot the values using the plot function.
62. Create a list having numbers in harmonic progression. Ask the user to enter the values of "a," "d," and the number of terms and plot the curve.
63. There are four types of parabolas: upward, downward, left facing, and rightfacing. The equations of the parabolas having vertex at the origin are as follows:
    Upward: x2 = 4ay
    Downward: x2 = – 4ay
    Right facing: y2 = 4ax
    Left facing: y2 = – 4ax
64. If the values of x range from [10, 10] and the values of y are calculated using the appropriate equation, (you can use comprehensions to calculate the values of y), plot the above parabolas on a single plot. Now create a subplot to plot each one of them.
65. Using plotting prove that (sin θ)2 + (cos θ)2 = 1.
66. Plot the curve of tan θ and identify the points of discontinuity.
67. Plot an ellipse having the length of major axis = 10 and the length of minor axis = 5.
68. Plot a circle having radius = 10 and center at (5,5).
69. Now plot 10 circles having radius 10 and center's x coordinate varying from 0 to 10. The y coordinate of the center should be 8.

Note: The examples provided are just a few, and additional ones can be included according to the syllabus.

Reference Books

1. Introduction to Computation and Programming Using Python: With Application to UnderstandingData, Guttag, John V. MIT Press.
2. Learn Python 3 the Hard Way: A Very Simple Introduction to the TerrifyinglyBeautiful World ofComputers and Code, Shaw, Zed A, Addison-WesleyProfessional.
3. Think Python 2e. Green Tea Books, Downey, Allen B.
4. Practical Programming: An Introduction to Computer Science Using Python 3.6. PragmaticBookshelf, Gries, Paul, Jennifer Campbell, and Jason Montojo.
5. Introduction to Python Programming, Gowrishankar S, Veena A, Talor and Francis.

6. Python® Programming for the Absolute Beginner, Third Edition, Michael Dawson, Cengage Learning.

CMAM - Theory: Mobile Application Development
SEC-3, Course, Theory, Semester – 3, Credits - 02, Contact hours - 30.

Mobile App development using Flutter and Dart

Introduction to Flutter Introducing Flutter, defining widgets and elements, understanding Widget lifecycle events, understanding the Widget tree and the element tree, Installing the Flutter SDK, **Android Setup:** Install Android Studio, Setup the Android Emulator.	02 hours
Creating Your First Flutter App Setting Up the Project, using hot reload, using themes to style your App, understanding Stateless and Stateful Widgets, using external packages.	02 hours
Learning Dart basic Purpose of DART and its use, Commenting code, Running the **main() entry Point**, referencing variables, declaring variables, using Operators, using flow statements, using functions, Import packages, using classes, Implementing Asynchronous Programming.	04 hours
Creating Starter Project Template Creating and Organizing Folders and Files, Structuring Widgets.	02 hours
Widget Tree Introduction to Widgets, Building the Full Widget Tree, Building a Shallow Widget Tree	02 hours
Using Common Widgets Using basic widgets, using Images and Icons, using decorators, using the Form Widget to validate text fields, checking orientation.	02 hours
User Interface (UI) Development **Animation:** Animated container, Crossfade, Opacity, controller. **Navigation:** Using navigator, Hero Animation, Bottom navigation bar, Bottom app bar, Tabbar and view. **Scrolling:** Card, list view, list tile, Gridview, using Stack. **Layout:** High level view of layout, Creating layout. **Interactivity:** Set up Gesture detector, Draggable and Dragtarget Widgets, Moving and Scaling, Dismissible Widget.	08 hours
Finalizing App development Understanding the JSON Format, Using Database Classes to Write, Read, and Serialize JSON, Formatting Dates, sorting a list of dates, Retrieving Data with the FutureBuilder, Building the Journal App, Adding the Journal Database Classes, Adding the Journal Entry Page, Finishing the Journal Home Page.	04 hours

Adding Firebase and Firestore Backend Introduction to Firebase and Cloud Firestore, Structuring and Data Modelling Cloud Firestore, Viewing Firebase Authentication Capabilities, Viewing Cloud Firestore Security Rules, Configuring the Firebase Project, adding a Cloud Firestore Database and Implementing Security, Building the Client Journal App, Adding Authentication and Cloud Firestore Packages to the Client App, Adding Basic Layout to the Client App, Adding Classes to the Client App.	04 hours

CMAM- Practical: Mobile App Development lab
SEC-3P, Practical, Semester – 3, Credits - 02, Contact hours - 45.

IDE: Android Studio, VS Code, UI Toolkit - Flutter.

Here are some practical assignments for mobile app development using Flutter;

1. Basic Flutter Mobile App
   - Create a simple Flutter app with a single screen that displays "Hello, Flutter!".
   - Add a button that changes the text to "Hello, World!" when pressed.

2. Personal Profile Mobile App
   - Develop an app that displays your personal profile, including your name, photo, and a brief bio.
   - Use different Flutter widgets such as `Container`, `Row`, `Column`, and `Text`.

3. Weather Mobile App
   - Develop a weather app that fetches and displays weather information for a given location.
   - Use an API like OpenWeatherMap or any other suitable and display the data using Flutter widgets.

4. Quiz Mobile App
   - Create a quiz app with multiple-choice questions.
   - Show the user's score at the end of the quiz.
   - Use `ListView` for displaying questions and options.

5. Photo Gallery Mobile App
   - Develop a photo gallery app that displays images from a user's device.
   - Implement features like viewing images in full screen, deleting, and sharing images.

Optional (App development for practice)

1. Simple Calculator
   - Build a basic calculator app that can perform addition, subtraction, multiplication, and division.
   - Use `TextField` for input and `RaisedButton` for operations.

2. Todo List App
   - Create a todo list app where users can add, edit, and delete tasks.
   - Use a `ListView` to display the list of tasks.

3. Recipe App
   - Create a recipe app that displays a list of recipes.
   - Each recipe should have a detail page with ingredients and instructions.
   - Implement navigation between the list and detail pages.

4. Notes App

- Build a notes app where users can create, view, edit, and delete notes.
- Use local storage (e.g., `SharedPreferences` or `sqflite`) to save the notes.

5. Expense Tracker
- Develop an expense tracker app that allows users to log their expenses.
- Display a summary of expenses by category and date.
- Use charts to visualize spending patterns.

6. E-commerce App
- Develop a simple e-commerce app with a product list and product detail pages.
- Implement a shopping cart where users can add products and proceed to checkout.

7. Chat App
- Build a basic chat application with a login screen and a chat screen.
- Use a backend service like Firebase for real-time messaging.

8. Fitness Tracker
- Create a fitness tracker app that logs workout activities.
- Display statistics like total time spent, calories burned, and workout history.

9. Music Player
- Create a music player app that can play audio files from the device.
- Implement basic controls like play, pause, next, and previous.

10. Travel Guide App
- Build a travel guide app that provides information about different travel destinations.
- Include features like a map view, destination details, and user reviews.

Reference books and other resources

1. Flutter Complete Reference by Alberto Miola
2. Beginning Flutter: A Hands-On Guide to App Development by Marco L. Napoli
3. Flutter in Action by Eric Windmill
4. Flutter for Beginners by Thomas Bailey and Alessandro Biessek
5. Pragmatic Flutter by Priyanka Tyagi
6. Online documentations by Google on Flutter: https://docs.flutter.dev/

Semester - IV				
Paper	**Paper type**	**Paper name**	**Credit**	**Contact hours**
DSCC-5	Theory	Multimedia and its application	3	45
	Practical	Multimedia and its application lab	1	30
DSCC-6	Theory	Computer Architecture & Organization	3	45
	Practical	Computer Architecture & OrganizationLab	1	30
DSCC-7	Theory	Database Management system	3	45
	Practical	RDBMS Lab	1	30
DSCC-8	Theory	Object Oriented Programming	3	45
	Practical	OOPs using Java	1	30

CMAM: Computer Application – Multimedia and its application
DSCC-5, Theory, Semester – 4, Credits - 03, Contact hours - 45.

Course Objective

The objective of the **"Multimedia and Its Applications"**course is to provide students with a thorough understanding of multimedia technologies and their practical applications across various domains. Students will learn to integrate and manipulate different media elements such as text, audio, images, video, and animation to create compelling and interactive content. The course will cover essential topics including Digital Data Acquisition, Media Representation and Media Formats, Colour Theory, Multimedia Authoring, Multimedia Compression, Application of Compression, Media Compression, Multimedia Distribution, and Wireless Multimedia Networking.

Syllabus Overview

1. Digital Data Acquisition: Techniques and tools for capturing and digitizing various forms of media.
2. Media Representation and Media Formats: Understanding different media formats and how they are represented digitally.
3. Colour Theory: Principles of colour and its application in multimedia design.
4. Multimedia Authoring: Tools and techniques for creating multimedia content.
5. Multimedia Compression: Methods for reducing the size of multimedia files without significant loss of quality.
6. Application of Compression: Practical uses of compression in various multimedia applications.
7. Media Compression: Advanced techniques for compressing different types of media.
8. Multimedia Distribution: Strategies for distributing multimedia content across different platforms.
9. Wireless Multimedia Networking: Technologies and protocols for delivering multimedia content over wireless networks.

Introduction to Multimedia **Introduction**: Components of multimedia; **History**: Past & present, hypermedia, WWW, Internet, multimedia in the new millennium; **Tools**: Audio, video, graphics, animation, authoring; **Future**: Trends in multimedia	**2**
Digital Data Acquisition	**4**

Signals: Analog vs digital, A/D conversion, sampling, quantization, bit rate; **Systems**: LTI systems, Fourier Transform basics; **Sampling**: Theorem, aliasing, Moiré patterns; **Filtering**: Digital filters, subsampling, Fourier theory	
Media Representation & Formats **Images**: Representation, aspect ratio, formats; **Video**: Analog vs digital, subsampling schemes, formats; **Audio**: Representation, surround/spatial audio, formats; **Graphics**: Overview	**4**
Colour Theory **Perception**: Human colour sensing & perception; **Trichromacity**: Cone response, tristimulus vector, calibration; **Colour Spaces**: RGB, CMYK, YUV, HSV, CIE XYZ; **Correction**: Gamma correction, monitor calibration	**3**
Multimedia Authoring **Tools**: Requirements & examples; **Intramedia**: Issues in images, video, audio, graphics; **Intermedia**: Spatial placement, temporal setup, interactivity; **Paradigms**: Timeline, scripting, flow control, cards; **Interfaces**: Mobile, multi-device, distributed authoring, content management	**5**
Multimedia Compression **Overview**: Need for compression; **Information Theory**: Entropy, efficiency; **Taxonomy**: Metrics, rate distortion; **Lossless**: RLE, Huffman, Arithmetic; **Lossy**: DPCM, transform coding, hybrid; **Issues**: Encoder speed, rate control, adaptive vs nonadaptive	**5**
Image Compression **Redundancy**: Image data; **Lossless**: GIF, PNG; **Transform Coding**: JPEG standard, bitstream, drawbacks; **Wavelet Coding**: JPEG 2000, preprocessing, DWT; **Transmission**: Progressive JPEG & JPEG 2000	**4**
Video Compression **Theory**: Temporal redundancy, motion vectors, macroblocks; **Prediction**: I, P, B frames, GOP structure; **Complexity**: Motion compensation methods; **Standards**: H.261, H.263, MPEG-1/2/4, H.264/AVC	**3**
Audio Compression **Theory**: Need, waveform coding (DPCM, ADPCM, quantization); **Structured Audio**: Event lists, synthesis; **Standards**: MPEG-1/2/4, Dolby AC-3, MP3, MIDI	**3**
Graphics Compression **Objects**: 2D points/curves, 3D polygons/patches; **Mesh Compression**: Triangle runs, topological surgery, progressive meshes; **Techniques**: Wavelet encoding, multiresolution; **Standards**: VRML, X3D, MPEG-4, Java 3D	**3**
Multimedia Distribution **Networking**: OSI, LAN/WAN; **Communication**: Unicast, multicast, broadcast; **Routing**: Algorithms, broadcast routing; **Traffic Control**: Congestion, flow control; **QoS**: Throughput, delay, error rate; **Protocols**: General & media-related	**4**
Wireless Multimedia Networking **Basics**: Wired vs wireless, RF spectrum, MAC protocols; **Standards**: Cellular, WLAN, Bluetooth; **Applications**: WAP; **Issues**: Multipath, attenuation, Doppler shift, handovers	**5**

CMAM: Multimedia and its application Lab
DSCC-5P, Practical, Semester – 4, Credits - 01, Contact hours - 30.

Here are some laboratory experiments related to multimedia and its applications that can considered for students:

Tools: GIMP, Audacity, Blender/Canva/OpenShot/Filmora, OBS, Synfig Studio.

1. Image Editing and Manipulation

- **Objective**: Learn the basics of image editing and manipulation.

- **Tools**: GIMP
- **Tasks**:
    - Adjust brightness, contrast, and colour balance.
    - Apply filters and effects.
    - Combine multiple images into a single composition.
- **Assessment Criteria**: Creativity, technical execution, and understanding of tools.

2. Audio Editing and Mixing

- **Objective**: Understand audio editing and mixing techniques.
- **Tools**: Audacity.
- **Tasks**:
    - Edit and trim audio clips.
    - Apply effects like reverb, echo, and equalization.
    - Mix multiple audio tracks.
- **Assessment Criteria**: Clarity, technical quality, and creativity.

3. Video Editing and Production

- **Objective**: Learn the fundamentals of video editing and production.
- **Tools**: Open shot.
- **Tasks**:
    - Import and organize video clips.
    - Apply transitions and effects.
    - Add titles, captions, and background music.
- **Assessment Criteria**: Storytelling, technical execution, and creativity.

4. 2D Animation

- **Objective**: Create a simple 2D animation.
- **Tools**: Synfig Studio.
- **Tasks**:
    - Develop a storyboard.
    - Design characters and backgrounds.
    - Animate scenes using keyframes.
- **Assessment Criteria**: Creativity, smoothness of animation, and technical execution.

5. 3D Modelling and Animation

- **Objective**: Understand the basics of 3D modelling and animation.
- **Tools**: Blender.
- **Tasks**:
    - Create 3D models of objects.
    - Apply textures and materials.
    - Animate the models.
- **Assessment Criteria**: Creativity, technical execution, and realism.

6. Broadcasting and Streaming

- **Objective**: Understand the basics of broadcasting, recording and streaming.
- **Tools**: OBS.
- **Tasks**:
    a. Screen recording.
    b. Broadcast and stream (YouTube live).

Assessment Criteria: Broadcasting with scene changing, filtering, recording screen content and YouTube live streaming.

These experiments can help students gain hands-on experience with various multimedia tools and techniques, enhancing their understanding and skills in multimedia applications.

Reference Books

1. Multimedia: Computing Communications & Applications, Klara Nahrstedt, Pearson India.
2. Multimedia and Applications, D.S. Sherawat, Sanjay Sharma, S.K. Kataria & Sons.
3. Introduction to Multimedia and Its Applications, V. K. Jain, Khanna Book Publishing Company.
4. Multimedia: Making it Work, Tay Vaughan, McGraw Hill Education.
5. Multimedia & Applications, Ashish Chopra, Ishan publications.
6. Principles of Multimedia, Ranjan Parekh, McGraw Hill Education.

CMAM: Computer Architecture & Organization
DSCC-6, Theory, Semester – 4, Credits - 03, Contact hours - 45.

Course Objective

The objective of the "Computer Organization and Architecture" course is to provide students with a comprehensive understanding of the fundamental principles and concepts underlying the design and functionality of computer systems. Students will explore the basic structure of computers, including the organization and design of the central processing unit (CPU), control unit, and memory. The course will cover essential topics such as Register Transfer and Micro-operations, CPU Registers, Instructions, and the differences between CISC and RISC processors. Additionally, students will learn about computer peripherals, input/output organization, and various memory types and management techniques.

Syllabus Overview

1. Basic Structure of Computers: Introduction to the fundamental components and their interactions.
2. Register Transfer and Micro-operation: Understanding data transfer and micro-operations within the CPU.
3. Basic Computer Organization and Design: Principles of computer design and organization.
4. CPU Organization: Detailed study of CPU architecture and its components.
5. Control Unit: Design and functionality of the control unit in a computer system.
6. CPU Registers: Types and roles of CPU registers in processing.
7. Instructions: Instruction set architecture and execution.
8. CISC and RISC Processors: Comparison and characteristics of CISC and RISC architectures.
9. Computer Peripherals: Overview of peripheral devices and their interfaces.
10. Input/Output Organization: Techniques and methods for I/O operations.
11. Memory: Types of memory, memory hierarchy, and management techniques.

Description	Contact hours
Basic Structure of Computers (Qualitative discussion) Basic functional units, basic operational concept, bus structure, software, performance, multiprocessor and multicomputer, IAS Computer, Historical Perspectives.	2 hours
Register Transfer and Micro-operation Register transfer language, register transfer, bus and memory transfers, three state bus buffers, memory transfer, arithmetic and logical micro-operations, shift and arithmetic shifts.	4 hours
Basic Computer Organization and Design Stored program organization, computer registers, common bus system, timing and	5 hours

control, instruction cycle, fetch decode, Computer Instructions, register reference instructions, memory reference instruction, input-output and Interrupt, design of basic computer, design of accumulator logic.	
CPU Organization Arithmetic and Logic Unit (ALU) - Combinational ALU, 2'S complement subtraction unit, Booth's algorithm for multiplication, restoration division algorithm and hardware. General register organization, control word, accumulator based, register based, Stack type CPU organization.	6 hours
Control Unit Hardwired Control Unit (basic concept), Micro-programmed Control Unit: Control memory, address sequencing, conditional branching, mapping of instructions, subroutine.	6 hours
CPU Registers Program Counter, Stack Pointer Register, Memory Address Register, Instruction Register, Memory Buffer Register, Flag registers, Temporary Registers.	4 hours
Instructions Operational code, operands, zero, one, two and three address instruction, instruction types, addressing modes, data transfer and manipulation instructions, Program control instructions.	5 hours
CISC and RISC processors Introduction, relative merits and De-merits.	1 hour
Computer Peripherals VDU, Keyboard, Mouse, Printer, Scanner (Qualitative approach).	3 hours
Input / Output Organization: Polling, Interrupts, subroutines, memory mapped I/O, I/O mapped IO, DMA, I/O bus and protocol, SCSI, PCI, USB, bus arbitration.	4 hours
Memory Primary memory: ROM, PROM, EPROM, EEPROM, Flash memory, SRAM, DRAM, **Cache Memory:** mapping functions, replacement algorithms, interleaving, hit and rate penalty, virtual memories, address translation, memory management requirements, **Secondary Storage:** Solid State drives (SSD), Magnetic hard disks, Optical disks, magnetic tape systems.	5 hours

CMAM: Computer Architecture & Organization
DSCC-6P, Semester – 4, Credits - 01, Contact hours - 30.

(1). Construct an Arithmetic Unit capable of performing 4-bit subtraction and Addition using 2's complement method. Use Parallel Adders and other necessary logic gates.
(2). Construct a 2-bit logical unit using logic gates capable of performing 2-bit, Bitwise ORing, ANDing, XORing and inversion
(3). Construct a 4-bit ALU unit which can perform the following operation;

Selection		Function
S_1	S_0	
0	0	Addition
0	1	Subtraction
1	0	XOR
1	1	Complement

(4). Construct a 2-bit **Carry Look Ahead (CLA)**Adder using logic gates.
(5). Study and construct a 1-digit BCD/Decimal adder using parallel adders and other necessary logic gates.

(6). Construct a Binary Multiplier using basic logic gates.
(7). Subtraction with 1's complement method using parallel adders and logic gates(necessary).
(8). Construction of BCD Subtractor with 9'S complement method using parallel adders.
(9). Construction of BCD Subtractor with 10'S complement method using parallel adders.
(10). Binary magnitude comparators (up to 4 bits) using parallel adder and logic gates.
(11). Cascading of 4-bit parallel adder (7483/74283) to construct an 8-bit adder circuit.
(12). Construct a Serial in Serial out 2/4-bit register.
(13). Construct a 2-bit Universal Shift register.
(14). Construct a 2/4-bit ring counter using edge triggered D Flip-Flops.
(15). Construct a 4 - bit Johnson Counter.
(16). Horizontal and Vertical Cascading of memory modules (7489/74189).
(17). Code converters using memory modules.

Text/Reference Books

1. Computer System Architecture, Morries Mano, Pearson.
2. Computer Organization & Architecture, Williams Stallings, Pearson.
3. Computer Organization, Hamacher, Vranesic and Zaky, McGraw Hill.
4. Computer Architecture and Organization, Govindrajalu, Tata McGraw Hill.
5. Computer Architecture and Organization, J P Hayes, Tata McGraw Hill.
6. Structured Computer Organization, Andrew S. Tanenbaum, Austin, Pearson.

Note: Laboratory work must be conducted using integrated circuits on a breadboard, along with other necessary devices and equipment.

CMAM: Database Management system
DSCC-7, Theory, Semester – 4, Credits - 03, Contact hours - 45.

Course Objective

The objective of this Database Management Systems (DBMS) course is to provide students with a comprehensive understanding of the fundamental concepts and techniques used in the design, implementation, and management of database systems.

- The course begins with an **Introduction** to database systems, covering their importance and applications.
- Students will then explore **Entity Relationship (ER) Modelling**, which is essential for conceptual database design.
- The **Relational Model** will be introduced, focusing on its structure and the principles behind it. **Integrity Constraints** will be discussed to ensure data accuracy and consistency.
- The course will delve into **Relational Database Design**, emphasizing normalization and schema refinement.
- Students will gain practical skills in **SQL** for querying and managing databases.
- Finally, the course will cover **Record Storage and File Organization** concepts, providing insights into how data is physically stored and accessed. By the end of the course, students will be equipped to design, implement, and manage efficient and effective database systems.

Description	Teaching Hours
Introduction Drawbacks of Legacy System; Advantages of DBMS; Layered Architecture of Database,Data Independence; Data Models; Schemas and Instances; Database Languages; DatabaseUsers, DBA; Data Dictionary.	04 hours
Entity Relationship (ER) Modelling Entity, Attributes and Relationship, Structural Constraints, Keys, ER Diagram of Some Example Database, Weak and strong Entity Set, Specialization and Generalization, Constraints of Specialization and Generalization, Aggregation.	04 hours
Relational Model Basic Concepts of Relational Model; Relational Algebra; Tuple Relational Calculus; Domain Relational Calculus.	08 hours
Integrity Constraints Domain Constraints, Referential Integrity, View.	04 hours
Relational Database Design Problems of Un-Normalized Database; Functional Dependencies (FD), Derivation Rules, Closure of FD Set, Canonical Cover; Normalization: Decomposition to 1NF, 2NF, 3NF or BCNF Using FD; Lossless Join Decomposition Algorithm; Dependency preservation.	10 hours
SQL Basic Structure, Data Definition, Constraints and Schema Changes; Basic SQL Queries (Selection, Insertion, Deletion, Update); Order by Clause; Complex Queries, Aggregate Function and Group by Clause; Nested Sub Queries; Views, Joined Relations; Set Comparisons (All, Some); Derived Relations.	10 hours
Record Storage and File Organization (Concepts only) Fixed Length and Variable Length Records; Spanned and Un-Spanned Organization of Records; Primary File Organizations and Access Structures Concepts; Unordered, Sequential, Hashed; Concepts of Primary and Secondary Index; Dense and Sparse Index; Index Sequential Files; Multilevel Indices.	05 hours

CMAM: RDBMS Lab

DSCC-7P, Theory, Semester – 4, Credits - 01, Contact hours - 30.

Assignments related to RDBMs using MYSQL

1. **Create a Database**:Design and create a database for a university, including tables for students, courses, and enrolments.
2. **Insert Data**:Populate the university database with sample data using INSERT INTO statements.
3. **Basic Queries**:Write SELECT queries to retrieve data from the students and courses tables.
4. **Filtering Data**:Use WHERE clauses to filter students based on their enrolment status or courses based on their credits.
5. **Aggregate Functions**:Calculate the average, minimum, and maximum grades for students using AVG, MIN, and MAX.
6. **Grouping Data**:Group students by their major and calculate the average GPA for each group using GROUP BY.
7. **Joining Tables**:Write JOIN queries to combine data from students and enrolments tables to list all students and their enrolled courses.
8. **Subqueries**:Use subqueries to find students who are enrolled in more than three courses.
9. **Updating Records**:Update the contact information for a student using the UPDATE statement.
10. **Deleting Records**:Remove students who have graduated using the DELETE statement.

11. **Creating Indexes**:Create indexes on the students table to improve query performance.
12. **Stored Procedures**:Write a stored procedure to calculate and update the GPA for each student.
13. **Triggers**:Create a trigger to automatically update the total number of students enrolled in a course when a new enrolment is added.
14. **Views**:Create a view to display the list of students along with their total credits earned.
15. **Transactions**:Implement transactions to ensure data integrity when enrolling students in courses.
16. **Normalization**:Normalize a given set of tables to the third normal form (3NF).
17. **Backup and Restore**:Perform a backup of the database and restore it to a new instance.
18. **User Management**:Create and manage user accounts with different privileges.
19. **Security**:Implement security measures such as encryption and access control.
20. **Performance Tuning**:Analyse and optimize query performance using EXPLAIN and other tools.

Note: The examples provided are just a few, and additional ones can be included according to the syllabus.

Text/ Reference Books

1. Fundamentals of Database Systems, R. Elmasri, S.B. Navathe, PearsonEducation.
2. Database Management Systems, R. Ramakrishanan, J. Gehrke,3rd Edition, McGraw-Hill.
3. Database System Concepts, A. Silberschatz, H.F. Korth, S. Sudarshan,McGraw Hill.
4. Database Systems Models, Languages, Design and application Programming, R. Elmasri,S.B. Navathe, Pearson Education.
5. SQL and Relational Theory: How to Write Accurate SQL Code, Christopher J. Date,O'Reilly Media.
6. Database Systems: A Practical Approach to Design, Implementation and Management,Thomas M.Connolly and Carolyn E. Begg, Pearson.

CMAM: Computer Application – Object Oriented Programming
DSCC-8, Theory, Semester – 4, Credits - 03, Contact hours - 45.

Course Objective

The objective of this Object-Oriented Programming (OOP) using Java course is to equip students with a solid foundation in the principles and practices of OOP, enabling them to design and develop robust, reusable, and maintainable software.

- The course will cover essential OOP concepts such as **abstraction**, **encapsulation**, **inheritance**, and **polymorphism**, and demonstrate their application in Java.
- Students will learn to create and manipulate classes and objects, implement interfaces, and use Java's built-in libraries effectively.
- The syllabus includes an **Introduction** to OOP and Java, **Classes and Objects**, **Methods and Constructors**, **Inheritance and Polymorphism**, **Exception Handling**, **File I/O**, and **Collections Framework**.
- By the end of the course, students will be proficient in using Java to solve complex programming problems and will have developed the skills necessary to build scalable and efficient software systems.

Description	Contact Hours
Concept of OOPs Difference with procedure-oriented programming, Data abstraction and information hiding: Objects, Classes, methods.	02 hours
Introduction to Java Java Architecture and features, understanding the semantic and syntax differences between C++ and Java, Compiling and executing a Java Program, variables, constants, keywords data types, Operators (Arithmetic, Logical and Bitwise) and expressions, comments, doing basic program output, decision making constructs (conditional statements and loops) and nesting, Java methods (defining, scope, passing and returning arguments, type conversion and type and checking, built-in Java class methods).	04 hours
Arrays, Strings and I/O Creating & using arrays (One dimension and multi-dimensional), referencing arrays dynamically, Java Strings: The Java String class, creating & using string objects, manipulating strings, string immutability & equality, passing strings to & from methods, string buffer classes. Simple I/O using System.out and the scanner class, byte and character streams, Reading/Writing from console and files.	06 hours
Object-Oriented Programming Overview Principles of Object-Oriented Programming, defining & using classes, controlling access to class members, class constructors, method overloading, Class variables & methods, Objects as parameters, final classes, Object class, garbage collection.	04 hours
Inheritance, Interfaces, Packages, Enumerations, Autoboxing and Metadata. Single Level and Multilevel, Method Overriding, Dynamic Method Dispatch, Abstract Classes, Interfaces and Packages, extending interfaces and packages, Package and Class Visibility, Using Standard Java Packages (util, lang, io, net), Wrapper Classes, Autoboxing/Unboxing, Enumerations and Metadata.	10 hours
Exception Handling, Threading, Networking and Database Connectivity Exception types, uncaught exceptions, throw, built-in exceptions, creating your own exceptions; Multi-threading: The Thread class and Runnable interface, creating single and multiple threads, Thread prioritization, synchronization and communication, suspending/resuming threads. Using java.net package, Overview of TCP/IP and Datagram programming. Accessing and manipulating databases using JDBC.	09 hours
Applets Java Applets: Introduction to Applets, Writing Java Applets, Working with Graphics, Incorporating Images & Sounds. Event Handling Mechanisms, Listener Interfaces, Adapter and Inner Classes. The design and Implementation of GUIs using the AWT controls, Swing components of Java Foundation Classes such as labels, buttons, textfields, layout managers, menus, events and listeners; Graphic objects for drawing figures such as lines, rectangles, ovals, using different fonts. Overview of servlets.	10 hours

CMAM - Practical: Object Oriented ProgrammingLab using JAVA
DSCC-8P, Semester – 4, Credits - 01, Contact hours - 30.

1. **Person Class:**Create a class called Person with attributes for name and age. Implement methods to set and get these attributes.
2. **Rectangle Class:**Create a class called Rectangle with attributes for width and height. Implement methods to calculate the area and perimeter.

3. **Circle Class:**Create a class called Circle with an attribute for radius. Implement methods to calculate the area and circumference.
4. **Book Class:**Create a class called Book with attributes for title, author, and ISBN. Implement methods to add and remove books from a collection.
5. **Employee Class:**Create a class called Employee with attributes for name, job title, and salary. Implement methods to calculate and update salary.
6. **Bank Account Class:**Create a class called BankAccount with attributes for account number, account holder's name, and balance. Include methods for depositing and withdrawing money, and checking the balance.
7. **Traffic Light Class:**Create a class called TrafficLight with attributes for colour and duration. Implement methods to change the color and check if the light is red or green.
8. **Inheritance Example:**Create a base class Shape with a method to calculate the area. Derive classes Triangle and Rectangle from Shape and override the area calculation method.
9. **Interface Example:**Create an interface Vehicle with methods for starting, stopping, and accelerating. Implement this interface in classes Car and Bike.
10. **Abstract Class Example:**Create -abstract class Animal with an abstract method makeSound(). Derive classes Dog and Cat from Animal and implement the makeSound() method.
11. **Polymorphism Example:**Create a class Shape with a method draw(). Derive classes Circle, Rectangle, and Triangle from Shape and override the draw() method. Demonstrate polymorphism by calling the draw() method on different shapes.
12. **Exception Handling:**Create a class that handles arithmetic exceptions. Implement methods to perform division and handle cases where division by zero occurs.
13. **File Handling:**Create a class to read from and write to a file. Implement methods to save and load a list of Person objects to and from a file.
14. **Collections Example:**Create a class to manage a collection of Book objects using an ArrayList. Implement methods to add, remove, and search for books.
15. **GUI Application:**Create a simple GUI application using Swing to manage a list of Employee objects. Implement features to add, remove, and display employees.
16. **ATM Simulation:**Create a class to simulate an ATM machine. Implement methods for checking balance, withdrawing money, and depositing money.
17. **Library Management System:**Create classes for Book, Member, and Library. Implement methods to issue and return books, and to manage the list of books and members.
18. **Shopping Cart:**Create classes for Product, CartItem, and ShoppingCart. Implement methods to add and remove items from the cart, and to calculate the total price.

Note: The examples provided are just a few, and additional ones can be included according to the syllabus.

Text/Reference Books

1. Java: The Complete Reference, Herbert Schildt, McGraw-Hill Education.
2. The Java Language Specification, Java SE by James Gosling, Bill Joy, Guy L Steele Jr, Gilad Bracha, Alex Buckley, Published by Addison Wesley.

3. Effective Java by Joshua Bloch, Publisher: Addison-Wesley.
4. Core Java 2 by Cay S. Horstmann, Gary Cornell, Volume 1, Prentice Hall.
5. Programming with Java by E. Balagurusamy, McGraw Hill.
6. Java: How to Program by Paul Deitel, Harvey Deitel, Prentice Hall.
7. Programming with JAVA by John R. Hubbard, Schaum's Series.

Semester - 5

Paper Code	Theory Paper	Credit	Contact hours	Practical/Tutorial Paper	Credit	Contact hours
DSCC-9	Data Communication and Networking	03	45	Computer Networking Lab	1	30
DSCC-10	Introduction to Algorithms	03	45	Algorithm Lab using C	1	30
DSCC-11	Operating System	03	45	Operating Systems Lab	1	30
DSCC-12	Mathematical Methods	03	45	Mathematical methods Lab using Python	1	30

DSCC-9;Theory: Data Communication and Networking
Credits - 03, Contact hours - 45.

Overview of Data Communication and Networking **Introduction:** Data communications Components, data representation, direction of data flow (simplex, half duplex, full duplex). **Network Hardware:** Physical structure (type of connection, topology), categories of network (LAN, MAN, WAN). **Internet:** Brief history, Protocols and standards, **Reference models:** OSI reference model, properties of all the layers, TCP/IP reference model, their comparative study.	03 hours
Physical Layer **Data & Signals:** Analog & Digital Data and Signals, periodic and non-periodic signals, composite signals, bandwidth, bit rate, transmission of digital signals. **Transmission Impairments:** Attenuation, Distortion and Noise. **Data Rate Limits:** Noiseless Channel: Nyquist Data rate, Noisy Channel: Shannon's Capacity, calculation of data rate using both limits. **Digital Transmission** **Digital to Digital Conversion:** Line coding, schemes (RZ, NRZ, Manchester, Differential Manchester), block coding. **Analog to Digital Conversion:** Sampling, Nyquist rate of sampling, Pulse code modulation (PCM), Delta Modulation (DM), Adaptive Delta Modulation (ADM), parallel and serial transmission. **Analog Transmission** **Digital to Analog conversion:** Amplitude shift keying (ASK), Frequency Shift Keying (FSK), Phase Shift Keying (PSK), Quadrature Amplitude Modulation (QAM). **Analog to Analog Conversion (qualitative):** Amplitude Modulation (AM), Frequency Modulation (FM), Phase Modulation.	11 hours
Bandwidth Utilization Techniques Multiplexing: FDM, Synchronous & Statistical TDM, WDM.	03 hours
Transmission Media **Guided media:** Twisted pair, Coaxial, Fiber optics. **Unguided:** Radio waves, microwaves, Infrared, Antenna, Communication satellites.	04 hours
Switching and Telephone network Circuit switched networks, Packet Switched networks, Virtual Circuit switch. Major components of telephone network, Dial up modem, DSL and ADSL modems, Cable TV	03 hours

for data transfer (qualitative study only)	
Data link Layer Types of errors, framing (character and bit stuffing), error detection & correction methods, Linear and cyclic codes, checksum. **Protocols:** Stop & wait ARQ, Go-Back- N ARQ, Selective repeat ARQ, HDLC (qualitative study only). **Physical addressing:** MAC address and its format	03 hours
Medium Access sub layer Point to Point Protocol, **Token Ring:** Reservation, Polling. **Multiple access protocols:** Pure & Slotted ALOHA, CSMA, CSMA/CD, CSMA/CA. **Channelization:** FDMA, TDMA, CDMA (Qualitative study only). **Wired and Wireless LAN:** Standards, fast Ethernet, Protocol 802.11, Bluetooth.	05 hours
Network layer **Internetworking & devices:** Repeaters, Hubs, Bridges, Switches, Router, Gateway, **Addressing:** IP addressing, Subnetting, Routing techniques: static vs. dynamic routing, **Protocols:** RARP, ARP, IP, ICMP.	07 hours
Transport layer **Process to Process delivery:** UDP, TCP	02 hours
Application Layer Introduction to DNS, Remote logging, FTP, Electronic mail, WWW & HTTP.	04 hours

Text/ Reference Books

1. Data Communication and Networking, B.A. Forouzan, Tata McGraw Hill.
2. Computer Networks, A.S. Tanenbaum, Pearson Education.
3. Data and Computer Communication, W. Stallings, Pearson Education.
4. Data & Computer Communication,Black, PHI.
5. Internet & World Wide Web: How to program,Harvey M. Deitel& Paul J. Deitel.

DSCC-9P;Practical:Networking Lab using CISCO PACKET TRACER
Credit: 01, Contact hour: 30.

1. Simulate Simplex, Half Duplex, and Full Duplex Communication Use serial and Ethernet ports to show directional data flow between two nodes.
2. Create LAN, MAN, and WAN Topologies Use appropriate devices (switches, routers, cloud) to build various category-wise networks.
3. Design Physical Topologies Implement star, bus, ring, and mesh topologies using appropriate devices.
4. Model Protocol Stack using TCP/IP Reference Model Layers Simulate communication between clients and servers, annotating each layer's activity.
5. Compare OSI vs TCP/IP Reference Models Create two scenarios showing layered encapsulation in both stacks and interpret packet flows.
6. Visualize Bandwidth and Bit Rate Concepts Use different cable types and configure link speeds to demonstrate throughput variations.
7. Simulate Signal Impairments Using Distorted Packets Introduce corruption manually and trace signal behavior across a noisy channel.
8. Compare Guided Media: Coaxial vs Fiber vs Twisted Pair Deploy multiple link types and measure latency and throughput across them.
9. Implement Multiplexing Models (FDM & TDM) Use time-sequenced communication across nodes to simulate multiplexed channel behavior.
10. Simulate Packet Switching Using Routers Design a scenario where data packets are rerouted based on availability.

11. Create a Virtual Circuit Setup Between Devices Use Label Switching or logical pathways to model a fixed route data exchange.
12. Demonstrate MAC Addressing and Frame Inspection Use simulation mode to trace MAC-level transmission and address resolution.
13. Simulate Error Detection Using Checksums and CRC Introduce errors and verify how they're flagged and corrected.
14. Frame Construction Using Bit and Character Stuffing Manual framing via scripts or event annotation showcasing stuffing techniques.
15. Compare CSMA/CD vs CSMA/CA Protocols Use wired Ethernet vs wireless 802.11 environments to simulate contention handling.
16. Create WLAN Using 802.11 Standard Devices Configure a wireless network with access points, laptops, and mobile nodes.
17. Simulate Bluetooth and PAN Communication Establish small-scale device-to-device networks to reflect low-energy protocol behavior.
18. Configure Static and Dynamic Routing Using RIP/EIGRP/OSPF Route between multiple routers with table inspection and simulation.
19. Implement IP Addressing and Subnetting Assign subnets to devices and verify connectivity across routed domains.
20. Simulate TCP and UDP Communication Use PCs and servers to demonstrate reliable vs unreliable transport with packet inspection.

Simulator:CISCO Packet Tracer

Note:*The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.*

DSCC-10; Theory: Introduction to Algorithms
Credits - 03, Contact hours - 45.

Introduction to Algorithms Definition, Characteristics, Recursive and Non-recursive algorithms	3 hours
Asymptotic Complexity Analysis of Algorithms Space and Time Complexity, Efficiency of an algorithm, Growth of Functions, Polynomialand Exponential Complexity, Asymptotic Notations: Big O Notation and Small o notation,Big Ω and Small ω, Big Θand Small φNotations, Properties: Best case/worst case/averagecase analysis of well-known algorithms.	6 hours
Algorithm Design Techniques Concepts and simple case studies of Greedy algorithms. Divide and conquer: Basic concepts, Case study of selected searching and sorting problems using divide and conquer techniques: Dynamic programming: General issues in Dynamic Programming.	12 hours
Graph Representation and Algorithm Graph traversal algorithms: BFS, DFS, Minimal spanning trees: Prim's Algorithm, Kruskal's Algorithm, Shortest path algorithms: Floyd's Algorithm, Floyd-Warshall Algorithm, Dijkstra's Algorithm, Graph Colouring Algorithms.	22 hours
Classification of Problems Concept of P, NP.	2 hours

Text/References Books

1. Introduction to Algorithms, Cormen, Leiserson, Rivest and Stein, TMH.
2. The Design and Analysis of Algorithms, Aho, Hopcroft and Ullman, Pearson Education.
3. The Art of Computer Programming, D.E. Knuth, Pearson Education.
4. Algorithm Design, Jon Kleiberg and Eva Tardos,Pearson Education.
5. Data Structures and Algorithms - K.Mehlhorn.
6. Computer Algorithms, S.Baase, Pearson Education.
7. Fundamentals of Computer Algorithms, E. Horowitz and Sahani, Galgotia
8. Combinational Algorithms- Theory and Practice, E.M. Reingold, J. Nievergelt and N. Deo,PHI.

DSCC-10P, Practical: Graph algorithms Lab using C
Credits - 01, Contact hours - 30.

Pre-requisite;

1. Programming Foundations in C
- Syntax, data types, loops, conditionals, and functions
- STL basics — especially vector, queue, stack, and map
- Dynamic memory and pointers (essential for building graph structures)

2. Graph Theory Fundamentals
- Terminology: vertices, edges, adjacency, degree, cycles
- Graph types: directed vs undirected, weighted vs unweighted
- Representations: adjacency matrix vs adjacency list

3. Math & Logic Readiness
- Basic combinatorics and logic reasoning
- Set theory (for understanding disjoint sets and connectivity)
- Matrices (helpful for adjacency matrix representations)

4. Compilers
- C compiler GCC

Laboratory exercises to be based on Graph Theory using C and based on the following;

Implementation of Graph algorithms: Single Spanning Tree Generation using - BFS, DFS, MinimalSpanning Tree Generation using - Prim's Algorithm, Kruskal's Algorithm, Shortest Path finding usingFloyd's Algorithm, Floyd-Warshall Algorithm, Dijkstra's Algorithm, Graph Partitioning Algorithm.

Sample questions

1. Write a C program to generate a single spanning tree using Breadth-First Search (BFS). Trace and display visited nodes.
2. Develop a Depth-First Search (DFS)-based C algorithm to construct a spanning tree from a connected undirected graph. Compare traversal order with BFS.
3. Design and implement Prim's Algorithm using adjacency list and priority queue in C. Display the edges included in the MST and their weights.
4. Construct Kruskal's Algorithm using disjoint set union in C. Identify cycle prevention strategy and display final edge set of the MST.
5. Write a C program to compute the shortest paths from a single source using Dijkstra's Algorithm. Provide output trace with distances and paths.
6. Implement Floyd's Algorithm to find all-pairs shortest paths in a weighted graph using adjacency matrix representation. Analyze time complexity.
7. Demonstrate Floyd-Warshall Algorithm with intermediate node storage. Display final distance matrix and reconstructed paths.

8. Apply a basic graph partitioning strategy using BFS clustering in C. Divide the graph into subgraphs and count nodes in each.
9. Implement a heuristic-based Graph Partitioning Algorithm in C and evaluate inter-partition edge cuts. Visualize the result if possible.

Note:*The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.*

DSCC-11, Theory: Operating System
Credits - 03, Contact hours - 45.

Introduction Basic OS functions, types of operating systems- batch processing, multiprogramming, timesharing, multiprocessing, distributed and real time systems.	5 hours
Operating System Organization Processor and user modes, kernels, system calls and system programs.	5 hours
Process System view of the process and resources, process control block, I/O and CPU boundprocess, process hierarchy, concept of threads. **Process Scheduling:**Pre-emptive and non-pre-emptive scheduling, Long term scheduling,short term/CPU scheduling (FCFS, SJF, SRJF, RR and priority) and medium-termscheduling. **Process Synchronization:** Concurrent processes, critical section, semaphores andapplication, methods for inter-process communication.	15 hours
Deadlock Definition, Prevention, Avoidance, Detection, Recovery.	8 hours
Memory Management Physical and logical address space; memory allocation strategies – fixed and variablepartitions, paging, segmentation, virtual memory.	7 hours
Memory Management Physical and logical address space, memory allocation strategies – fixed and variablepartitions, paging, segmentation, virtual memory.	3 hours
Protection and Security Policy mechanism, Authentication.	2 hours

Text/ Reference Books

1. Operating Systems Concepts, A Silberschatz, P.B. Galvin, G. Gagne, John Wiley Publications.
2. Modern Operating Systems, A.S. Tanenbaum, 3rd Edition, Pearson Education.
3. Operating Systems: A Modern Perspective, G. Nutt, Pearson Education.
4. Operating Systems, Internals & Design Principles W.Stallings, PHI.
5. Operating Systems- Concepts and design, M. Milenkovic, Tata McGraw Hill.
6. UNIX Concepts and Applications, Sumitabha Das, Tata McGraw-Hill.
7. Understanding the Linux Kernel, D. P. Bovet and M. Cesati, O'Reilly.

DSCC-11P, Practical: Operating System Lab (Shell Programming)
Credits - 01, Contact hours - 30.

1. Write a shell script to convert the content of a file from lower case to upper case.

2. Write a shell script to count the words, lines and characters of a given file. File name should beprovided at run time.
3. Write a shell script that take a word from user and find out the frequency of the word in a givenfile.
4. Write a shell script that gets executed at the moment of user login and it displays Good Morning,Good afternoon, Good Evening, Good Night, depending upon the time at which the user logs on.
5. Write a shell script to print Pascal diamond.
6. Write a shell script to find a number using sequential search method.
7. Write a shell script to find a number using binary search technique.
8. Write a shell script to sort a set of integer numbers using bubble sort.
9. Write a shell script to find out the factorial of a given number.
10. Write a shell script to reverse a string and check whether it is a palindrome.
11. Write a shell script to find the roots of a quadratic equation ax2 + bx +c = 0, considering allpossible cases.
12. Write a shell script for menu-based system to insert records for employees with employee ID, name, designation, salary in a data file, also display records when necessary. Display salary forthe employee asked.

Note:*The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.*

DSCC-12, Theory: Mathematics methods
Credits - 03, Contact hours - 45.

Introduction **Set Theory:** Finite and Infinite Sets, Uncountable Infinite Sets, Relations: Properties ofBinary Relations, Closure, Partial Ordering Relations, Equivalence, Functions: definition, one-to-one, onto and invertible, Mathematical Functions: Exponential and Logarithmic,**Counting:** Mathematical Induction, Pigeonhole Principle, Permutation and Combination, Binomial Theorem, Principle of Inclusion and Exclusion.	08 hours
Introduction to Probability Elementary events, Sample space, Classical and Axiomatic definition of Probability,Theorems on Total Probability, Conditional Probability, Bernoulli Trials and BinomialDistribution, Bayes' Theorem, Random Variables, Expectation, Variance, StandardDeviation.	07 hours
Growth of Functions Asymptotic Notations, Standard notations and common functions with simple examples.	03 hours
Recurrences Relations, Generating Functions, Linear Recurrence Relations with Constant Coefficients and their solution, Substitution Method, Recurrence Trees.	04 hours
Numerical Methods (Algorithmic Approach) **Errors:** Approximate and Rounding of Numbers, Significant digits, Errors and their types, Propagation of errors. **Interpolation:** Newton Forward and Backward interpolation, Lagrange interpolation. **Solving a Set of Linear Equations:** Gaussian Elimination, Gauss–Jordan, Iteration methods and their convergence conditions, Gauss-Seidel, Gauss-Jacobi Iterative	

Methods. Solving Non-linear equations: Bisection, Regula-falsi, Secant and Newton-Raphson, their order of convergence. **Solving Differential Equations:** Euler, Runge-Kutta second and fourth order methods. Numerical Integration: Trapezoidal and Simpson's 1/3rd rules. **Curve fitting:**Least square approximation, Linear regression, Polynomial regression, Fitting Exponentialand Trigonometric functions.	14 hours
Graph Theory Basic Terminology, Models and Types, Multi graphs and Weighted graphs, GraphRepresentation, Graph Isomorphism, Connectivity, Euler and Hamiltonian Paths andCircuits, Planar Graphs, Trees and their basic terminologies and properties.	09 hours

Text/ Reference Books

1. Elements of Discrete mathematics, C.L. Liu & Mahapatra, Tata McGraw Hill.
2. Discrete Mathematics and Its Applications, Rosen, McGraw Hill.
3. Introduction to algorithms, T.H. Cormen, C.E. Leiserson, R. L. Rivest, Prentice Hall.
4. Discrete Mathematics with Algorithms, M. O. Albertson and J. P. Hutchinson, JohnWiley Publication.
5. Discrete Structures, Logic, and Computability, J. L. Hein, Jones and Bartlett Publishers.
6. Essentials of Discrete Mathematics, D.J. Hunter, Jones and Bartlett Publishers.
7. Numerical Analysis and Computational Procedures by Mollah, New Central Book.
8. Computer Oriented Numerical Methods, 3rd Edition, V Rajaraman, PHI
9. Graph Theory with Applications to Engineering and Computer Science by NarsinghDeo, PHI.
10. Graph Theory by J.A. Bondy and U.S.R. Murty, Springer.
11. Introduction to Graph Theory by D B West, 2nd edition, Pearson Education

DSCC-12P, Practical: Mathematical Methods Lab using Python.
Credits - 01, Contact hours - 30.

1. Set Operations: Implement union, intersection, difference, and symmetric difference on finite sets using Python's set data type.
2. Countable vs Uncountable Sets: Simulate countable sets (e.g., integers) and discuss uncountable sets (e.g., real numbers) using generator functions.
3. Binary Relations: Represent relations as sets of ordered pairs and check properties: reflexivity, symmetry, transitivity.
4. Equivalence and Partial Order: Identify equivalence classes and Hasse diagrams using adjacency matrices.
5. Function Properties: Write Python functions to test one-to-one, onto, and invertibility using mappings.
6. Sample Space Generator: Create sample spaces for coin tosses, dice rolls, and card draws.
7. Classical Probability Calculator: Compute probabilities using combinatorics (e.g., probability of drawing a red card).
8. Bayes' Theorem Simulation: Implement conditional probability and Bayes' rule with real-world examples.
9. Bernoulli Trials and Binomial Distribution: Simulate trials and plot binomial distribution using matplotlib.
10. Random Variables and Expectation: Generate discrete random variables and compute expectation, variance, and standard deviation.
11. **Asymptotic Notation Visualizer**: Plot functions like n, nlogn, n^2 2^n to compare growth rates.
12. **Recurrence Solver**: Implement substitution and recurrence tree methods to solve recurrence relations.

13. **Generating Functions**: Use symbolic computation (sympy) to derive generating functions for sequences.
14. **Error Analysis**: Compute absolute, relative, and percentage errors for approximated values.
15. **Interpolation Techniques**: Implement Newton Forward/Backward and Lagrange interpolation using tabular data.
16. **Solving Linear Systems**: Code Gaussian Elimination, Gauss-Jordan, and iterative methods (Jacobi, Gauss-Seidel).
17. **Root-Finding Algorithms**: Implement Bisection, Regula-Falsi, Secant, and Newton-Raphson methods with convergence checks.
18. **ODE Solvers**: Apply Euler and Runge-Kutta (2nd and 4th order) methods to solve first-order differential equations.
19. **Numerical Integration**: Implement Trapezoidal and Simpson's 1/3rd rule for definite integrals.
20. **Curve Fitting**: Use least squares to fit linear, polynomial, exponential, and trigonometric curves to data.
21. **Graph Representation**: Build graphs using adjacency lists and matrices; visualize with networkx.
22. **Graph Traversals**: Implement BFS and DFS; trace visited nodes and paths.
23. **Euler and Hamiltonian Paths**: Check for Eulerian and Hamiltonian circuits in undirected graphs.
24. **Planar Graph Checker**: Use Kuratowski's theorem heuristics to test planarity.
25. **Tree Properties**: Implement tree traversal algorithms and check properties like height, leaf count, and degree.

Software tool/Compiler:"Miniconda or Anaconda can be used to set up the Python environment, and Jupyter Notebook can be used for Python programming.

Note:*The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.*

Semester - 6

Paper Code	Theory Paper	Credit	Contact hours	Practical/Tutorial Paper	Credit	Contact hours
DSCC-13	Introduction to Machine Learning.	03	45	Machine learning Lab	1	30
DSCC-14	Embedded Systems	03	45	Embedded System Labs	1	30
DSCC-15	Software Engineering	03	45	Software Engineering Lab	1	30

DSCC-13, Theory: Introduction to Machine Learning
Credits - 03, Contact hours - 45.

Introduction to Machine Learning Definition and scope of ML, Types of learning: Supervised, Unsupervised, Reinforcement, Applications in computer vision, NLP, and business analytics, Python libraries: NumPy, Pandas, Scikit-learn.	06 hours
Data Preprocessing and Feature Engineering Data cleaning, normalization, encoding, Handling missing values and outliers, Feature selection and dimensionality reduction, Introduction to PCA.	07 hours
Supervised Learning Algorithms Linear Regression and Logistic Regression, Decision Trees and Random Forests, Support Vector Machines (SVM), k-Nearest Neighbours (k-NN), Evaluation metrics: Accuracy, Precision, Recall, F1-score.	11 hours
Unsupervised Learning Algorithms Clustering: k-Means, Hierarchical, Association rule mining: Apriori algorithm, Dimensionality reduction: PCA, t-SNE.	07 hours
Probability and Statistical Foundations Probability distributions: Normal, Binomial, Poisson, Bayes' Theorem and Naive Bayes Classifier, Expectation, variance, and standard deviation, Hypothesis testing basics.	07 hours
Neural Networks and Deep Learning (Introductory) Perceptron and multilayer networks, Activation functions, Introduction to TensorFlow/Keras, Simple feedforward network for classification.	07 hours

Text/ Reference Books

1. Introduction to Computation and Programming Using Python: With Application to UnderstandingData, Guttag, John V. MIT Press.
2. Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World ofComputers and Code, Shaw, Zed A, Addison-Wesley Professional.
3. Think Python 2e. Green Tea Books, Downey, Allen B.
4. Practical Programming: An Introduction to Computer Science Using Python 3.6. PragmaticBookshelf, Gries, Paul, Jennifer Campbell, and Jason Montojo.

5. Introduction to Python Programming, Gowrishankar S, Veena A, Taylor and Francis, CRC Press.
6. Python Cookbook, David Beazley and Brian K. Jones, O'Reilly.

DSCC-13P, Practical: Machine Learning Lab using Python
Credits - 01, Contact hours - 30.

Lab 1: Data Exploration and Visualization
Objective: Load a dataset, perform basic statistics, and visualize features. Task:

- Load the Iris dataset using sklearn.datasets
- Use Pandas to display summary statistics
- Plot histograms, boxplots, and scatter plots using Matplotlib/Seaborn Outcome: Understand data distribution and feature relationships.

Lab 2: Linear Regression
Objective: Predict continuous values using regression. Task:

- Use the Boston Housing dataset
- Train a Linear Regression model using sklearn.linear_model
- Evaluate using Mean Squared Error and R^2 score Outcome: Learn regression modelling and performance metrics.

Lab 3: Logistic Regression for Classification
Objective: Perform binary classification. Task:

- Use the Titanic dataset (Kaggle)
- Predict survival using logistic regression
- Evaluate with confusion matrix and ROC curve Outcome: Understand classification and model evaluation.

Lab 4: Decision Trees and Random Forests
Objective: Apply tree-based models for classification. Task:

- Use the Iris or Loan Prediction dataset
- Train Decision Tree and Random Forest models
- Visualize the tree structure Outcome: Compare tree-based classifiers and interpret decisions.

Lab 5: k-Nearest Neighbours (k-NN)
Objective: Implement instance-based learning. Task:

- Use the Wine dataset
- Train a k-NN classifier and tune k
- Evaluate using accuracy and confusion matrix Outcome: Learn distance-based classification.

Lab 6: k-Means Clustering
Objective: Perform unsupervised clustering. Task:

- Use the Mall Customer Segmentation dataset
- Apply k-Means and visualize clusters
- Use Elbow method to choose optimal k Outcome: Understand clustering and intra-cluster variance.

Lab 7: Principal Component Analysis (PCA)
Objective: Reduce dimensionality and visualize data. Task:

- Use the Breast Cancer dataset
- Apply PCA and reduce to 2D
- Plot principal components Outcome: Learn feature reduction and variance preservation.

Lab 8: Naive Bayes Classification

Objective: Apply probabilistic classification. Task:

- Use SMS Spam Collection dataset
- Preprocess text and train Naive Bayes model
- Evaluate precision and recall Outcome: Understand Bayes' theorem in classification.

Lab 9: Neural Network with Keras

Objective: Build a simple feedforward neural network. Task:

- Use MNIST or Fashion-MNIST dataset
- Train a model using TensorFlow/Keras
- Plot training accuracy and loss Outcome: Learn basics of deep learning and model tuning.

Lab 10: Mini Project

Objective: Apply end-to-end ML pipeline. Task:

- Choose a dataset (e.g., student performance, diabetes prediction)
- Perform preprocessing, model training, evaluation, and visualization Outcome: Demonstrate complete ML workflow and reporting.

Software Tool/Compiler: - Python (Miniconda/Anaconda)

Note:*The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.*

DSCC-14, Theory: Embedded Systems
Credits - 03, Contact hours - 45.

Introduction to Embedded Systems Definition, characteristics, and applications, Embedded vs general-purpose systems, Microcontrollers vs microprocessors, Real-time systems and constraints.	**02 hours**
Microcontroller Architecture & Programming 8-bit (8051), Memory organization (Data and Program memory), Special function registers, I/O port registers and architecture, Instruction set, timers, interrupts, Serial communications (qualitative only), Assembly Language programming.	**10 hours**
Arduino Platform & Interfacing Arduino architecture and toolchain, Programming essentials, Digital and analog I/O, PWM, serial communication, **Sensor interfacing:**temperature, humidity, pressure, soil moisture motion, light, gas sensors, location sensors (GPS), display interfacing (LED, LCD and DOT matrix), **Actuator control:**motors, buzzers.	**09 hours**
Raspberry Pi & Python Programming Introduction to Raspberry Pi architecture(qualitative) and GPIO access, Raspbian Operating system (Installation and basic GPIO configuration), Python scripting, Interfacing sensors and actuators via GPIO, introduction to Node-Red.	**07 hours**
Communication Protocols & IoT Integration Basics of UART, SPI, I2C, **Wireless protocols:**Bluetooth, Wi-Fi, ZigBee, Cloud connectivity and MQTT basics. Sending sensor data to cloud using Raspberry Pi or ESP8266, use of NODE-RED for interfacing and IoT.	**06 hours**
Embedded System Design & Applications	

Design flow:requirement → architecture → implementation, **Case studies:**Smart agriculture, wearable health monitors, Smart home automation, Smart Farming, Smart grid, and Smart cities.	**04 hours**
Embedded AI Fundamentals (Introductory) Concept: What is Embedded AI, Edge vs Cloud: Local vs remote AI processing, Toolchain: Overview of STM32CubeMX and STM32Cube.AI.	**03 hours**
Neural Networks for Embedded Use (Elementary) **Basics**: Layers, activation, inference,**Model Creation**: Simple overview with Python/Keras,**Optimization**: Quantization for low-power devices.	**04 hours**
AI Deployment with STM32 (Elementary) **Deployment Flow**: Steps to convert trained models into embedded code; **Use Cases**: Gesture recognition, sensor-based classification.	**05 hours**

References:

1. An Embedded software primer,David E.Simon, Pearson Education.
2. The 8051 Microcontroller, Kenneth J. Ayala, Thomson.
3. Embedded Systems, Raj Kamal, TMH.
4. Microcontroller, Raj Kamal, Pearson Education.
5. **The 8051 Microcontroller Based Embedded Systems,** Manish K. Patel, McGraw Hill Education (India).
6. The 8051 Microcontroller and Embedded Systems, Muhammad Ali Mazidi.
7. STM32F103 Arm Microcontroller and Embedded Systems, Sarmad Naimi, Muhammad Ali Mazidi, Sepehr Naimi.
8. **Embedded Systems and IoT,** Dr. Ratna Kamala Petla, Cengage India.
9. **Embedded System Design,** Santanu Chattopadhyay, PHI Learning.
10. TM32 Embedded Systems Design: Definitive Reference for Developers andEngineers, Richard Johnson, HiTeX Press; PublishDrive edition.
11. Online resources from ST electronics: https://wiki.st.com/stm32mcu/wiki/STM32StepByStep:Getting_started_with_STM32_:_STM32_step_by_step.

DSCC-14P, Practical: Embedded System Lab
Credits - 01, Contact hours - 30.

Laboratory Assignments on 8951 series microcontrollers using Keil

1. Copy a block of 20 bytes of data available from address 60H to 73H to the location starting from 40H.
2. Shift a block of 8 bytes of data, presently located from 50H to 57H, 1 byte up, so that the data is available from 51H to 58H.
3. Twenty bytes of data are stored in location from 7FH to 6CH of internal RAM. Count the number of those bytes, which contain 00H, and store this number of null bytes in RAM location 6BH.
4. Sixteen consecutive bytes starting from 50H have unsigned integers. Develop a program to add all these 16 integers and store the 8-bit sum in memory location 60H.
5. Write a program to generate and store natural numbers starting from 1 to ‘N’ terms and also find the sum of these numbers. Assume that the value of ‘N’ is stored in location 30H. Store generated natural numbers from 40H. Leave the sum in the accumulator.
6. Write a program to find the sum of the series 1 − 2 + 3 − 4 + … up to N terms. Assume the non-zero value of N is available in location 30H. Store the sum in the accumulator.

7. Generate Fibonacci series up to N terms and save from location 50H onwards. Assume N being available in location 4FH. Also assume the value of N to be greater than 3.
8. Generate and store the following series up to N terms. Value of N is available in location 30H. The series is presented using decimal number system. 1, 2, 3, 11, 12, 13, 21, 22, 23, 31, … up to N terms.
9. Two arrays, each of 16 bytes, are stored from 40H to 4FH and 50H to 5FH. Write a program to compare these two arrays and store 00H in location 30H if they are identical (same number and in same sequence). Otherwise, store a non-zero value in location 30H.
10. Sixteen random numbers are stored in location 40H to 4FH. Write a program to take out alternate numbers, starting from the first number, and create a second array of eight numbers from 50H to 57H.
11. Extend the previous program so that the first array is compacted within 8 bytes from 40H to 47H after shifting eight of its entries.
12. Sixteen random numbers are stored in an array, starting from location 40H. Write a program to count the number of non-zero elements in this array and store it in location 30H.
13. Find out number of 1s in a byte, available in internal data memory location 30H. Store the result in the accumulator itself.
14. A packed BCD number is in location 30H. After unpacking, store it in R6 (lower digit) and R7 (higher digit).
15. Two unpacked BCD digits are available in location 30H (MS digit) and 31H (LS digit). Write a program to pack these two and store in R7.
16. Using only one pointer, R0, pack two arrays of BCD digits to create a third array. Th e higher digits are available from 30H to 3FH. Lower digits are available from 40H to 4FH. Packed BCD numbers are to be stored from 50H to 5FH.
17. Find the largest and smallest from N unsigned integers. Assume the value of N to be available in the internal data memory location 30H. The array starts from location 31H. Store the maximum integer in R4 and minimum in R3.
18. A few unsigned integers are stored from the location 31H onwards of the internal data memory area. Arrange these integers in ascending order. Th e number of terms of the array is available in the location 30H. Store the arranged integers from 31H itself.
19. A few random unsigned integers are stored from the internal data memory location 31H onwards. Number of terms (N) is available in location 30H. Assuming that none of these numbers is greater than 5, find the factorials of these integers and then find their sum. Assume that the sum would not exceed 8-bit value.
20. Create a new array by removing only those integers that are perfectly divisible by 4 from an array, starting from 31H. Location 30H contains number of terms of this array. Th e new array is to be created from the location 60H. At return, the accumulator should indicate number of terms found. Original locations with digits divisible by 4 should be replaced by null.
21. Register R4 would be loaded by an unsigned integer, which may vary from 0 to 5. Find the square of the integer and store it in R5.
22. Sixteen-bit values of the square of integers are stored in a table that starts from program memory
23. location 0200H onwards. The lower byte of the 16-bit number is at the lower address, and the higher byte is at the higher address. Assuming that the accumulator contains the integer whose square is required, write a routine to get the result using table look-up procedure (MOVC instruction.

Laboratory Assignments on Arduino

24. Develop a program to generate dynamic LED patterns using Arduino’s digital output pins and timed logic.

25. Interface a 16×2 LCD with Arduino to display real-time messages or sensor data in formatted text. 26. Use a dot matrix display to scroll custom messages by interfacing it with Arduino and controlling animation frames.
26. Read pressure values from a BMP180 or BMP280 sensor and display the readings using Arduino's serial monitor or LCD.
27. Measure ambient temperature and humidity using a DHT11 sensor and display the results on an LCD.
28. Detect human motion using a PIR sensor and trigger an LED or buzzer response through Arduino.
29. Measure the distance of nearby objects using the HC-SR04 ultrasonic sensor and display the result on a screen.
30. Detect proximity using an IR sensor and activate an output device such as an LED or buzzer when an object is nearby.
31. Interface a GPS module with Arduino to read and display latitude and longitude coordinates in real time.
32. Measure orientation and acceleration using the MPU6050 sensor and process the data through Arduino.
33. Detect ambient sound levels using a microphone sensor and trigger alerts when the sound exceeds a threshold.
34. Monitor air quality by interfacing an MQ-series gas sensor with Arduino and display gas concentration levels.
35. Use a soil moisture sensor to monitor plant health and trigger a visual or audible alert when moisture drops below a set level.
36. Control the angle of a servo motor using a potentiometer input and map the analog value to servo rotation.
37. Interface a stepper motor with Arduino and control its rotation direction and step count for positioning applications.

Laboratory Assignments using Raspberry Pi, NODE-RED and Python

38. Use Node-RED to control an LED connected to a GPIO pin and toggle its state via a dashboard button.
39. Write a Python script to blink an LED at regular intervals using GPIO output control.
40. Interface a push button with Raspberry Pi and use Node-RED to display its pressed/released status on a dashboard.
41. Create a Python program to read digital input from a push button and toggle an LED accordingly.
42. Use Node-RED to read temperature and humidity data from a DHT11 sensor and visualize it using a gauge widget.
43. Write a Python script to interface a PIR sensor and trigger a buzzer when motion is detected.
44. Interface an ultrasonic sensor using Python to measure distance and display the result on the terminal.
45. Use Node-RED to read distance data from an HC-SR04 sensor and plot it on a real-time chart.
46. Create a Python program to read analog light intensity using an LDR and an ADC module, then display the value.
47. Use Node-RED to monitor soil moisture levels and trigger an alert when the value drops below a threshold.
48. Interface a gas sensor using Python and display air quality status based on analog readings.
49. Use Node-RED to read GPS coordinates from a serial-connected GPS module and display location data on a map.
50. Write a Python script to read acceleration and orientation data from an MPU6050 sensor and log it to a CSV file.
51. Use Node-RED to control a servo motor's angle based on slider input from the dashboard.

52. Create a Python program to rotate a stepper motor in both directions using GPIO pins and delay logic.

Laboratory Assignments using STM32

53. Write a program to blink an LED connected to a GPIO pin using STM32's internal delay functions.
54. Configure a push button as digital input and toggle an LED each time the button is pressed.
55. Use Timer 2 to generate a precise delay and blink an LED at a fixed interval.
56. Interface a 16×2 LCD with STM32 and display a static message using GPIO and delay logic.
57. Read analog voltage from a potentiometer using ADC and display the value on the serial monitor.
58. Generate PWM signals using Timer 3 and control the brightness of an LED.
59. Interface a DHT11 sensor and display temperature and humidity readings via UART.
60. Measure distance using an HC-SR04 ultrasonic sensor and display the result on an LCD.
61. Interface a servo motor and control its angle using PWM output from STM32.
62. Configure external interrupt on a GPIO pin to trigger an LED when a falling edge is detected.

Tools/Simulator/Programming Language: PYTHON, Rasbian, Node-Red(Pre installed in Rasbian OS of Raspberry Pi).

Development board/single board-Development board based on 8951/52, Arduino Uno/MEGA, Single board computers: Raspberry Pi 3+ or higher and STM32.

Sensors required: Ultrasonic sensor (HC-SR04), PIR Sensor, IR Proximity Sensor, **Acceleration and tilt sensing and Gyroscope:** ADXL345, GPS Module (NEO-6M), Bluetooth (HC-05/HC-06), Wi-Fi (ESP8266/ESP32), LoRa Module, DHT11/DHT22, BMP180/BMP280, LM35/DS18B20, Air quality sensor-MQ 135. LCD 16 x 2, DOT Matrix display, and LED etc.

Miscellaneous: Battery, adaptors, connecting wires, and breadboard etc.

Note:*The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.*

DSCC-15, Theory: Software Engineering
Credits - 03, Contact hours - 45.

Introduction Defining system, open and closed system, modelling of system through computerhardware, communication systems, external agents and software systems, Importance ofEngineering Methodology towards computerization of a system.	3 hours
Software Life Cycle Classical and Iterative Waterfall Model, Spiral Model; Prototype Model, Evolutionary model and its importance towards application for different system representations, Comparative Studies.	6 hours
Software Requirement and Specification Analysis Requirements Principles and its analysis principles, Specification Principles and itsRepresentationsSoftware Design Analysis – Different level of DFD Design, Physical and Logical DFD,Use and Conversions between them, Decision Tables and Trees, Structured analysis,Coupling and Cohesion of different modulesSoftware Cost Estimation Modelling – COCOMO.	15 hours
Software Testing Software Verification and Validation; Testing objectives, Testing Principles, Testability,Error and Faults, Unit Testing, White Box and Blank Box Testing, Test case	15 hours

design:Test Vector, Test Stub.	
Software Quality Assurances Concepts of Quality, Quality Control, Quality Assurance, IEEE Standard for Statistical Software Quality Assurances (SSQA) criterions.	06 hours

Text/References Books

1. Software Engineering: A Practitioner's Approach by R.S. Pressman, McGraw-Hill.
2. An Integrated Approach to Software Engineering by P. Jalote, Narosa Publishing House.
3. Software Engineering by K.K. Aggarwal and Y. Singh, New Age International Publishers.
4. Software Engineering by I. Sommerville, Addison Wesle.
5. Software Engineering for Students by D. Bell, Addison-Wesley.
6. Fundamentals of Software Engineering by R. Mall, PHI.

DSCC-15P, Practical: Software Engineering Lab
Credits - 01, Contact hours - 30.

Lab 1: Getting Started with Gaphor.
Objective: Familiarize students with Gaphor's interface and basic UML elements Tasks:

- Install Gaphor via pip install gaphor
- Create a new model and diagram
- Add basic elements: Class, Interface, Package
- Save and export the diagram as PNG Outcome: Students understand the modelling environment and diagram setup.

Lab 2: Class Diagram – Library Management System
Objective: Model object-oriented relationships **Tasks**:

- Create classes: Book, Member, Librarian, Transaction
- Define attributes and methods
- Show associations, aggregations, and multiplicity **Outcome**: Students grasp encapsulation, inheritance, and object interaction.

Lab 3: Use Case Diagram – Online Shopping Portal
Objective: Capture functional requirements **Tasks**:

- Identify actors: Customer, Admin, Payment Gateway
- Define use cases: Browse, Order, Pay, Track
- Connect actors to use cases **Outcome**: Students visualize system behaviour from user perspectives

Lab 4: Activity Diagram – ATM Withdrawal Process
Objective: Model control flow and decision logic **Tasks**:

- Create an activity diagram for Withdraw Cash
- Include decision nodes, actions, and swimlanes
- Annotate with guard conditions **Outcome**: Students understand process modelling and branching logic

Lab 5: State Machine Diagram – Traffic Light Controller
Objective: Model reactive systems Tasks:

- Define states: Red, Green, Yellow
- Add transitions and events
- Simulate timing and control logic Outcome: Students explore state-based behavior and event-driven design.

Lab 6: Component Diagram – Web Application Architecture
Objective: Visualize modular system design **Tasks**:

- Identify components: Frontend, Backend, Database, API
- Show dependencies and interfaces
- Annotate with protocols (e.g., HTTP, SQL) **Outcome**: Students learn modular design and deployment structure.

Lab 7: Package Diagram – Python Project Structure
Objective: Organize codebase and modules **Tasks**:

- Create packages: utils, models, controllers, views
- Show dependencies and containment
- Reflect on Python file structure **Outcome**: Students connect UML with actual Python project organization.

Software Tool/Compiler: Gaphor.

Note:*The questions provided are illustrative samples. Students are encouraged to practice additional exercises aligned with the prescribed syllabus topics for comprehensive understanding.*

Semester - 7

Paper Code	Theory Paper	Credit	Contact hours	Practical/Tutorial Paper	Credit	Contact hours
DSCC-16	Cryptography and Cyber Security	03	45	Cryptography and Cyber Security lab using Python	1	30
Internship		16	12 Weeks			
The industrial internship program to be organized in collaboration with partner institutes and industry organizations. It should be designed to provide students with hands-on exposure to both hardware and software domains, ensuring that the core areas of study in Computer Applications are effectively covered. Through this partnership, students will gain practical experience in real-world environments, bridging theoretical knowledge with industry practices and enhancing employability.						

DSCC-16, Theory: Cryptography and Cyber Security
Credits - 03, Contact hours - 45.

Introduction to Security Concepts Need for security, Security approaches, CIA Triad (Confidentiality, Integrity, Availability),types of attacks (malware, phishing, DoS, insider threats),case Studies of breaches.	6 hours
Fundamentals of Cryptography Plain text & Cipher text; Substitution & Transposition techniques, Encryption & Decryption, Symmetric vs Asymmetric Cryptography, Steganography, Key range & Size,attack types.	8hours
Symmetric Key Algorithms Overview & Modes, DES; IDEA, RC5, Blowfish, AES, Differential & Linear Cryptanalysis.	8hours
Asymmetric Key Algorithms RSA Algorithm, Hybrid Cryptography (Symmetric + Asymmetric), Digital Signatures (MD, MD5, SHA, HMAC), Knapsack Algorithm, Other Algorithms, Attacks on Digital Signatures.	8hours
Public Key Infrastructure (PKI) Introduction to PKI, Digital Certificates; Private Key Management, PKIX Model, PKCS Standards, XML PKI & Security.	5hours
Internet Security Protocols SSL/TLS in TCP/IP; SHTTP, Time Stamping Protocol (TSP), SET, Email Security (qualitative), GSM Security, WEP & WPA/WPS.	5hours
User Authentication Mechanisms Authentication Basics, Passwords, Tokens, Certificate-based authentication, Biometrics, Kerberos, Single Sign-On (SSO).	3hours
Governance & Ethics	2hours

Cyber Laws & Compliance (GDPR, IT Act), Ethical Hacking & Pen testing overview, Security Policies & Risk Management.	

DSCC-16P, Practical: Cryptography and Cyber Security Lab using Python
Credits - 01, Contact hours - 30.

(The assignments listed are illustrative examples; additional exercises may be incorporated in alignment with the syllabus and expanded further for practice and examination purposes).

1. Write a Python program to implement Caesar Cipher encryption and decryption for a given text and key.
2. Design a Python script that performs monoalphabetic substitution cipher using a predefined key mapping.
3. Implement columnar transposition encryption and decryption in Python for a given message.
4. Write a Python program to generate MD5 and SHA-256 hashes for user-entered strings.
5. Develop a Python script to simulate brute-force password cracking on a weak hashed password.
6. Use Python's cryptography library to encrypt and decrypt a text file using AES.
7. Write a Python program to generate RSA keys and demonstrate encryption and decryption of a message.
8. Implement digital signing and verification of a message using RSA and SHA hashing.
9. Create a Python program to generate and verify HMAC for message integrity.
10. Write a Python script to hide a secret message inside an image using the Least Significant Bit (LSB) method.
11. Develop a Python socket program to send an encrypted file securely between two systems.
12. Use PyShark or Scapy in Python to capture network packets and analyse suspicious traffic.
13. Write a Python script using socket to scan open ports on a given host.
14. Create a simple Flask web app with a vulnerable login form and demonstrate SQL injection attacks.
15. Write a Python program to parse system log files and detect failed login attempts.
16. Develop a Python script to scan files for known malicious hashes.
17. Implement frequency analysis in Python to break a substitution cipher.
18. Write a Python script to scrape CVE vulnerability data from a security website.
19. Simulate secure communication between IoT devices using encrypted MQTT messages in Python.
20. Implement password hashing with bcrypt and salting in Python for secure storage.

Suggested readings

1. Cryptography and Network Securityby Atul Kahate, McGraw Hill.
2. Network Security and Cryptography by William Stallings, Pearson.
3. Python for Cybersecurity Cookbookby Nishant Krishna, bpb publication.
4. Cryptography And Information Securityby Pachghare V. K, PHI.
5. Cryptography and Network Securityby Rathnakar Achary, New Age Publications.
6. Introduction To Cryptography And Network Security by Behrouz A. Forouzan, McGraw Hill.
7. Cryptography And Network Security (Sie), Forouzan,McGraw-Hill Education (India) Pvt Limited.

Semester - 8

Paper	Credit
Project	16
Projects may be undertaken in both hardware and software domains, aligned with the prescribed syllabus. Each student is required to carry out the project during the semester and present the completed work for evaluation in the examinations. The project must include either a working model or a functional prototype, accompanied by appropriate documentation in the form of a structured report.	