VIJAYANAGARA SRI KRISHNADEVARAYA UNIVERSITY

JNANASAGARA CAMPUS, BALLARI-583105

Department of Studies in

Bachelor of Application

I & II Semester Syllabus

BACHELOR OF BACHELOR OF APPLICATION

Programme as per State Education Policy 2024

Under Choice Based Credit System (CBCS)

With effect from 2024-25 and onwards

Title of Subject: PROGRAMMING IN C (Major – 1)

COURSE CODE: 24MJBCA1L1	**CIA Marks: 20**	
SEMESTER: I	**SEE Marks: 80**	
Contact Hours: (L:T:P): 4-0-0	**Credit: 04**	**Duration of Exam: 03**

Course Outcomes: After completing this course satisfactorily, a student will be able to:

1. Write algorithms, flowcharts for given problems
2. Read, understand and trace the execution of programs written in C language and Write the C code for a given problem
3. Implement different programming constructs and decomposition of problems into functions.
4. Use and implement data structures like arrays and structures to obtain solutions.
5. Define and use of pointers with simple programs.

UNIT – I:	**10 Hours**

Problem Solving: Algorithm, characteristics of algorithm; Flowchart: understand the different symbols used to design flowcharts. Draw flowchart for the problem.

Introduction to C Programming: Overview of C; History and Features of C; Structure of a C Program with Examples; Creating and Executing a C Program; Compilation process in C.

C Programming Basic Concepts: C Character Set; C tokens - keywords, identifiers, constants, and variables; Data types; Declaration & initialization of variables; Symbolic constants.

UNIT – II:	**12 Hours**

Input and output with C: Formatted I/O functions - printf and scanf, control stings and escape sequences, output specifications with printf functions; Unformatted I/O functions to read and display single character and a string - getchar, putchar, gets and puts functions.

C Operators & Expressions: Arithmetic operators; Relational operators; Logical operators; Assignment operators; Increment & Decrement operators; Bitwise operators; Conditional operator; Special operators; Expressions, Types of Expressions, Operator Precedence and Associativity, Evaluation of arithmetic expressions; Type conversion, Mathematical functions.

UNIT – III:	**12 Hours**

Control Structures: Decision making Statements - Simple_if, if_else, nested if_else, else_if ladder, Switch Case, goto. Looping Statements - Entry controlled and exit controlled statements, while, do-while, for loops, Nested loops, break & continue statements;

Arrays: Arrays: One Dimensional arrays - Declaration, Initialization and Memory representation; Two Dimensional arrays - Declaration, Initialization and Memory representation, Processing Arrays.

Strings: Declaring & Initializing string variables; String handling functions - strlen, strcmp, strcpy and strcat.

UNIT – IV:	**10 Hours**

User Defined Functions: Need for user defined functions; Format of C user defined functions; Components of user defined functions - return type, name, parameter list, function body, return statement and function call; Categories of user defined functions - With and without parameters and return type.

Pointers in C: Understanding pointers - Declaring and initializing pointers, accessing address and value of variables using pointers; Pointers and Arrays; Pointer Arithmetic; Advantages and disadvantages of using pointers;

UNIT – V:	**12 Hours**

User defined data types: Structures - Structure Definition, Advantages of Structure, declaring structure variables, accessing structure members, Structure members initialization, comparing structure variables, Array of Structures, Embedded structures; Unions - Union definition, defining a union, declaring union variables, accessing union members, union members initialization; difference between Structures and Unions.

File management in C: Introduction, Defining and opening a file, closing a file, Input/output and Error Handling on Files

Text Books:

1. Balaguruswarny: Programming in ANSI C, Tata McGraw-Hill.
2. Brian W. Kernighan and Dennis M. Ritchie: The C Programming Language, PHI

Reference:

1. V. Rajaraman: Fundamentals of Computers, PHI(EEE).
2. Kamthane, Programming with ANSI and Turbo C. Pearson Education, Asia.
3. Herbert Schildt: C. The complete reference, 4th edition.
4. Yeshwant Kanetkar: Let us C, BPB Publications.
5. Rajesh Hongal Computer Concepts and C Programming.

Title of Subject: C PROGRAMMING LAB (Major – 1)		
COURSE CODE: 24MJBCA1P1	CIA Marks: 10	
SEMESTER: I	SEE Marks: 40	
Contact Hours: (L:T:P): 0-0-4	Credit: 02	Duration of Exam: 03

Part A

Algorithm and Flowchart

- **Algorithm Writing**: Write algorithms for basic problems such as finding the largest of three numbers, calculating the factorial of a number, and generating Fibonacci series.
- **Flowchart Design**: Draw flowcharts for the above algorithms using standard symbols.

Introduction to C Programming

- **Simple C Program**: Write a program to display "Hello, World!" and understand the structure of a C program.
- **C Program Structure**: Write a program to add two numbers and display the result. Discuss the structure of the C program (including headers, main function, and return statement).

C Programming Basic Concepts

- **Character Set and Tokens**: Write a program to demonstrate the use of keywords, identifiers, constants, and variables.
- **Data Types**: Write a program to declare and initialize variables of different data types (int, float, char, double).
- **Symbolic Constants**: Write a program to define and use symbolic constants.

Formatted I/O Functions

- **printf and scanf**: Write a program to read and display an integer, a float, and a character using printf and scanf.
- **Control Strings and Escape Sequences**: Write a program to demonstrate the use of different control strings and escape sequences in printf and scanf.

Unformatted I/O Functions

- **Single Character I/O**: Write a program to read and display a single character using getchar and putchar.
- **String I/O**: Write a program to read and display a string using gets and puts.

C Operators and Expressions

- **Arithmetic Operators**: Write a program to perform basic arithmetic operations (+, -, *, /, %).
- **Relational and Logical Operators**: Write a program to demonstrate the use of relational and logical operators.
- **Assignment and Increment/Decrement Operators**: Write a program to demonstrate the use of assignment, increment, and decrement operators.

- **Bitwise and Conditional Operators**: Write a program to demonstrate the use of bitwise and conditional operators.

Decision Making Statements

- **if-else Statements**: Write a program to find the largest of three numbers using if-else statements.
- **Nested if-else and else-if Ladder**: Write a program to classify a student's grade based on marks using nested if-else and else-if ladder.
- **Switch Case**: Write a program to print the day of the week using switch-case statements.

Part B

Looping Statements

- **while Loop**: Write a program to print the first 10 natural numbers using a while loop.
- **do-while Loop**: Write a program to print the factorial of a number using a do-while loop.
- **for Loop**: Write a program to generate the Fibonacci series using a for loop.
- **Nested Loops**: Write a program to print a multiplication table using nested loops.

Arrays

- **One Dimensional Arrays**: Write a program to read and display elements of a one-dimensional array.
- **Two Dimensional Arrays**: Write a program to read and display elements of a two-dimensional array.
- **Array Processing**: Write a program to find the sum and average of elements in an array.

Strings

- **String Handling Functions**: Write a program to demonstrate the use of string handling functions (strlen, strcmp, strcpy, strcat).

User Defined Functions

- **Simple Function**: Write a program to find the square of a number using a user-defined function.
- **Functions with Parameters**: Write a program to calculate the sum of two numbers using a function with parameters and return type.
- **Functions without Parameters**: Write a program to display a message using a function without parameters and return type.

Pointers in C

- **Pointer Basics**: Write a program to declare and initialize pointers and access the value and address of variables using pointers.
- **Pointers and Arrays**: Write a program to demonstrate the relationship between pointers and arrays.
- **Pointer Arithmetic**: Write a program to perform pointer arithmetic (increment, decrement, addition, subtraction).

Structures

- **Structure Definition**: Write a program to define a structure for a student (roll number, name, marks) and demonstrate initialization and access of structure members.
- **Array of Structures**: Write a program to create an array of structures for students and display their details.
- **Embedded Structures**: Write a program to demonstrate embedded structures (structure within a structure).

Unions

- **Union Basics**: Write a program to define a union for different data types (int, float, char) and demonstrate initialization and access of union members.
- **Difference between Structures and Unions**: Write a program to show the memory usage difference between structures and unions.

File Management in C

- **File Operations**: Write a program to create, open, read, write, and close a file.
- **Error Handling in File Operations**: Write a program to handle errors during file operations (e.g., file not found, read/write errors).

Evaluation Scheme for Lab Examination:

Assessment Criteria		**Marks**
Program – 1 from Part A	Writing the Program	07
	Execution	08
Program -2 from Part B	Writing the Program	07
	Execution	08
Practical Record		05
Viva-Voce		05
Total		**40**

<table>
<tr><th colspan="3">Title of Subject: DIGITAL LOGIC & COMPUTER DESIGN (Major – 2)</th></tr>
<tr><td><b>COURSE CODE: 24MJBCA1L2</b></td><td colspan="2"><b>CIA Marks: 20</b></td></tr>
<tr><td><b>SEMESTER: I</b></td><td colspan="2"><b>SEE Marks: 80</b></td></tr>
<tr><td><b>Contact Hours: (L:T:P): 4-0-0</b></td><td><b>Credit: 04</b></td><td><b>Duration of Exam: 03</b></td></tr>
<tr><td colspan="3"><b>Course Outcomes:</b> After completing this course satisfactorily, a student will be able to:
1. CO1: Understand significance of number systems, conversions, binary codes
2. CO2: Apply different simplification methods for minimizing Boolean functions
3. CO3: Illustrate knowledge on design of various combinational circuits
4. CO4: Illustrate the concept of sequential logic design, analyze the operation of flip-flops, registers, and counters</td></tr>
<tr><td colspan="2"><b>UNIT – I:</b></td><td><b>10 Hours</b></td></tr>
<tr><td colspan="3"><b>Number system and codes:</b> Number Systems: Binary number system, decimal number system, octal number system, hexadecimal number system. conversion from one number system to another. Complement representation of negative numbers: Signed Magnitude, One’s complement method, Two’s complement method, Binary Arithmetic. Codes: BCD, GRAY, EXCESS-3, ASCII and Unicode, error detection and error correction codes.</td></tr>
<tr><td colspan="2"><b>UNIT – II:</b></td><td><b>12 Hours</b></td></tr>
<tr><td colspan="3"><b>Boolean algebra and logic systems</b>: Laws of Boolean algebra, Boolean laws. Evaluation of Boolean expression, De Morgan's theorems and proof, simplification on Boolean expressions using Boolean laws. Basic gates (AND, OR, NOT): truth table, Definition, Boolean expression and symbols, universal gates (NAND, NOR): truth table, definition, Boolean expression and symbols, design of basic gates using NAND and NOR gates- Logical gates using NAND and NOR, Design of given Boolean expression using basic gates or universal gates. XOR and XNOR gates (Definition, Boolean expression and symbols, truth table).</td></tr>
<tr><td colspan="2"><b>UNIT – III:</b></td><td><b>12 Hours</b></td></tr>
<tr><td colspan="3"><b>Simplification of Boolean functions:</b> SOP and POS form, min term and max term, expression of Boolean equation in Min and Max term (conversion of SOP and POS forms to standard form) K-map method: Rules, simplification of Boolean equation using K-map (up to 4 variables), without and with don't-care condition, Implementation using basic gates or NAND gate or NOR gate, Quine - McCluskey Tabulation method, determination and selection of prime implicates</td></tr>
<tr><td colspan="2"><b>UNIT – IV:</b></td><td><b>10 Hours</b></td></tr>
<tr><td colspan="3"><b>Combination logic:</b> Design procedure, design of half adder and full adder, half subtract or and full sub tractor. Code converters: - BCD to Excess 3 code, gray code, magnitude comparator, encoders (BCD to decimal), decoder (decimal to BCD), Multiplexer (4:1 and 8:1) DeMultiplexer(1:4 and 1:8).</td></tr>
<tr><td colspan="2"><b>UNIT – V:</b></td><td><b>12 Hours</b></td></tr>
<tr><td colspan="3"><b>Sequential logic:</b> Introduction, Flip-flops — SR, JK, D, T, JK-MS (Detailed Study) Registers — Introduction, shift register- types and applications. Counters — synchronous and asynchronous counters (Up, down, up down).</td></tr>
<tr><td colspan="3"><b><u>Text Book:</u></b>
1. M- Moris Mano, Computer System Architecture, 2 Edition, Prentice Hall of India.</td></tr>
</table>

Reference:

1. Heuring and Jordan, Computer systems design and architecture, Pearson Education
2. William Stallings, Computer Organization and Architecture, Pearson Education 2003.
3. Andrew S Tenenbaum, Structured Computer Organization, 3rd Edition, Prentice Hall of India(1990).
4. Kamthane, Programming with ANSI and Turbo C. Pearson Education, Asia.
5. Herbert Schildt: C. The complete reference, 4th edition.
6. Yeshwant Kanetkar: Let us C, BPB Publications.
7. Rajesh Hongal Computer Concepts and C Programming.

Title of Subject: DIGITAL LOGIC LAB (Major – 2)		
COURSE CODE: 24MJBCA1P2	CIA Marks: 10	
SEMESTER: I	SEE Marks: 40	
Contact Hours: (L:T:P): 0-0-4	Credit: 02	Duration of Exam: 03

Part A

Number System Conversion

Binary to Decimal Conversion:

- Create a simple binary to decimal conversion circuit using binary switches (inputs) and a binary-to-decimal display component.
- Use combinational logic gates to add the binary inputs and display the decimal result.

Decimal to Binary Conversion:

- Use a decimal switch as input and connect it to a binary output display using binary-to-decimal encoder.
- Implement a logic circuit that converts decimal input to binary output.

Octal to Hexadecimal Conversion:

- Create a circuit with octal input switches and connect them to a hexadecimal display.
- Use intermediate binary representation for conversion.

Complement Representation

Signed Magnitude Representation:

- Design a circuit that takes a binary number and a sign bit as input.
- Use logic gates to display the signed magnitude representation.

One's Complement Method:

- Create a circuit that computes the one's complement of a binary number.
- Use NOT gates to invert each bit of the input number.

Two's Complement Method:

- Design a circuit that computes the two's complement by inverting the bits (using NOT gates) and adding one (using a binary adder).

Binary Arithmetic

Binary Addition:

- Implement a binary adder circuit using half adders and full adders.
- Use binary switches for input and an LED display for output.

Binary Subtraction:

- Design a circuit to perform binary subtraction using two's complement method and a binary adder.

Code Conversions

BCD to Binary Conversion:

- Use BCD switches as input and connect them to a binary output display using a BCD-to-binary encoder circuit.

Binary to Gray Code Conversion:

- Design a circuit that converts a binary number to its Gray code equivalent using XOR gates.

Gray Code to Binary Conversion:

- Implement a circuit that converts Gray code input back to binary using XOR gates.

ASCII Conversion:

- Create a circuit that displays the ASCII value of a given character and vice versa.
- Use switches for input and a seven-segment display for output.

Boolean Expression Evaluation

Boolean Laws:

- Design circuits to verify Boolean laws using basic logic gates.
- Use switches as inputs and LEDs to display the results.

De Morgan's Theorems:

- Implement circuits to verify De Morgan’s theorems using NOT, AND, and OR gates.

- Display the results using LEDs.

Basic Gates

AND, OR, NOT Gates:

- Create separate circuits for AND, OR, and NOT gates.
- Use truth tables to test the functionality.

NAND and NOR Gates as Universal Gates:

- Design circuits to implement AND, OR, and NOT gates using only NAND and NOR gates.

XOR and XNOR Gates

XOR Gate:

- Implement an XOR gate circuit using AND, OR, and NOT gates.
- Display the output using LEDs.

XNOR Gate:

- Implement an XNOR gate circuit using AND, OR, and NOT gates.
- Display the output using LEDs.

Part B

SOP and POS Forms

SOP to POS Conversion:

- Create a circuit that converts SOP form to POS form using logic gates.
- Test with sample inputs and verify the outputs.

POS to SOP Conversion:

- Create a circuit that converts POS form to SOP form using logic gates.
- Test with sample inputs and verify the outputs.

Half Adder and Full Adder

Half Adder Design:

- Implement a half adder circuit using XOR and AND gates.
- Use switches for input and LEDs for output.

Full Adder Design:

- Design a full adder circuit using two half adders and an OR gate.
- Use switches for input and LEDs for output.

Half Subtractor and Full Subtractor

Half Subtractor Design:

- Implement a half subtractor circuit using XOR and AND gates.
- Use switches for input and LEDs for output.

Full Subtractor Design:

- Design a full subtractor circuit using two half subtractors and an OR gate.
- Use switches for input and LEDs for output.

Code Converters

BCD to Excess-3 Code Converter:

- Design a circuit to convert BCD input to Excess-3 code using logic gates.
- Use switches for input and LEDs for output.

Gray Code Converter:

- Implement a circuit to convert binary to Gray code using XOR gates.

Encoders and Decoders

BCD to Decimal Encoder:

- Design a BCD to decimal encoder circuit using logic gates.
- Use switches for input and LEDs for output.

Decimal to BCD Decoder:

- Implement a decimal to BCD decoder circuit using logic gates.

Multiplexers and Demultiplexers

4:1 Multiplexer:

- Design a 4:1 multiplexer circuit using AND, OR, and NOT gates.
- Use switches for input and LEDs for output.

1:4 Demultiplexer:

- Implement a 1:4 demultiplexer circuit using AND, OR, and NOT gates.

Flip-Flops

SR Flip-Flop:

- Design an SR flip-flop circuit using NAND gates.
- Use switches for input and LEDs for output.

JK Flip-Flop:

- Implement a JK flip-flop circuit using NAND gates.
- Use switches for input and LEDs for output.

D Flip-Flop:

- Design a D flip-flop circuit using NAND gates.
- Use switches for input and LEDs for output.

T Flip-Flop:

- Implement a T flip-flop circuit using NAND gates.
- Use switches for input and LEDs for output.

JK-MS Flip-Flop:

- Design a JK master-slave flip-flop circuit using NAND gates.

Implementation Steps Using Logisim

1. **Install Logisim**: Download and install Logisim from the official website.
2. **Create New Project**: Start Logisim and create a new project.
3. **Design Circuit**: Use the toolbar to select components (gates, switches, LEDs, etc.) and design the circuit as per the program requirements.
4. **Simulate**: After designing the circuit, simulate it by providing inputs using switches and observing outputs on LEDs or other display components.
5. **Save and Document**: Save the circuit design and document the steps taken, including screenshots and descriptions for each part of the circuit.

Evaluation Scheme for Lab Examination:

Assessment Criteria		Marks
Program – 1 from Part A	Writing the Program	07
	Execution	08

Program -2 from Part B	Writing the Program	07
	Execution	08
Practical Record		05
Viva-Voce		05
Total		**40**

Title of Subject: Mathematics (Major – 3)

COURSE CODE: 24MJBCA1L3	CIA Marks: 20	
SEMESTER: II	SEE Marks: 80	
Contact Hours: (L:T:P): 4-0-0	Credit: 04	Duration of Exam: 03

Course Outcomes:

- Study and solve problems related to connectives, predicates, and quantifiers under different situations.
- Develop basic knowledge of matrices and to solve equations using Cramer’s rule. Know the concept of Eigen values.
- To develop the knowledge about derivatives and know various applications of differentiation.
- Verify trigonometric identities, using proper logic and use trigonometric identities to evaluate expressions.
- Solve trigonometric equations.
- Take limits of algebraic and trigonometric expressions of the form 0/0 (that’s simplify), non-zero number over 0, including limits that go to (positive or negative) infinity, limits that don't exist and limits that are finite.
- Differentiate and integrate all polynomial, rational, and trigonometric functions, and compositions of those functions.

UNIT – I:	10 Hours

Mathematical logic: Introduction-statements Connectives-negation, conjunction, disjunction-statement formulas and truth tables- Conditional and Biconditional statements- tautology and contradiction- Quantifiers, negation, consequences of implication-contrapositive and converse, problems, proving a statement by the method of contradiction by giving counter example.

UNIT – II:	10 Hours

Matrix algebra: Introduction-Types of matrices-matrix operations- transpose of a matrix-determinant of matrix- inverse of a matrix-Cramer’s rule, finding rank of a matrix - normal form-echelon form, Eigen values and Eigen vectors, Cayley Hamilton theorem-Eigen values

UNIT – III:	12 Hours

Trigonometry: Trigonometric functions, Measuring angles in radians and indegrees and conversion from one measure to another. Definition of trigonometric functions with the help of unit circle. Truth of the identity’ sin2x + cos2 = 1, for all x. Expressing sin (x+ y) and cos (x + y) in terms of sin x, sin y, cos x and cos y and their simple applications. Definition of allied angles and obtaining theirtrigonometric ratios using compound angle formulae. Identities related to sin2x, cos2x, tan2x, sin3x, cos3x and tan3x.

UNIT – IV:	12 Hours

Differential Calculus: Functions and limits, Continuity of a function, Fixed point property of continuous function, Differentiability - Simple Differentiation of Algebraic Functions, product rule and quotient rule– Evaluation of First and Second Order Derivatives.

UNIT – V:	12 Hours

Integral Calculus: Definition, Indefinite nature of integration, standard elementary integrals, Integration by substitution, examples. Integration using trigonometric identities, examples. Integration by partial fractions and Integration by parts (simple problems), Definite Integrals and Properties of definite integrals.

Text Books:

1. Discrete Mathematics 2nd Edn. (Schaum's Outline Series), Seymour Lipschutz, Marc Lipson, Tata Mc-Graw Hill.
2. Shanti Narayan, A Text book of Matrices, S. Chand Publishing N. Delhi
3. S. L Loney, Plane Trigonometry, Part1 and 2, Arihant Publications,2016
4. Shanti Narayan, Integral Calculus, S. Chan & Co.1999.
5. Shanti Narayan, Differential Calculus, S. Chand & Co.1998.

References:

1. M. Shantakumar, Engineering Mathematics–Volume I, Vasundhara Publishers, Mysore.
2. Dr.B.S. Grewal, Elementary Engineering Mathematics, Khanna Publishers, Delhi.
3. H K Das. Advanced Engineering Mathematics, S. Chand & Co., N. Delhi.2019.

Title of Subject: Accountancy (Major – 3)

COURSE CODE: 24MJBCA1L3	**CIA Marks: 20**	
SEMESTER: II	**SEE Marks: 80**	
Contact Hours: (L:T:P): 4-0-0	**Credit: 04**	**Duration of Exam: 03**

Course Outcomes:

- Study and understand Accounting, systems of Book, Branches of accounting advantage and limitations.
- Know the concept of accounting, financial and accounting process and Journalization.
- Maintenance different account book and reconciliations
- Preparations of different bills, and trial balance.

UNIT – I: 12 Hours

Introduction: History and Development of Accounting, Meaning, Objectives and functions of Accounting, Bookkeeping V/s Accounting, Users of accounting data, systems of book keeping and accounting, branches of accounting, advantages and limitations of accounting

UNIT – II: 10 Hours

Accounting Concepts and Convention: Meaning, need and classification, accounting standards meaning, need and classification of Indian accounting standards.

Financial Accounting Process: Classification of accounting transactions and accounts, rules of debit and credit as per Double Entry System. Journalizing and Ledger posting.

UNIT – III: 10 Hours

Preparation of Different Subsidiary Books: Purchase Day book Sales Day Book, Purchase Returns Day Book, Sales Returns Day Book, Cash Book. Bank Reconciliation Statement: Meaning, Causes of Difference, Advantages, Preparation of Bank Reconciliation Statements

UNIT – IV: 12 Hours

Account Procedure: Honor of the Bill, Dishonor of the Dill, Endorsement, Discounting, Renewal, and Bill for collection, Retirement of the Bill, Accommodation Bills, Bill Receivable Book and Payable Book. Preparation of Trial Balance: Rectification of errors and Journal Proper

UNIT – V: 12 Hours

Preparation of Final Accounts: Meaning, need and classification, Preparation of Profit and loss account and Balance – Sheet of sale- traders and partnership firms.

Text Books:

1. S. Ramesh, B.S. Chandrashekar, A Text Book of Accountancy.
2. V.A. Patil and J.S. Korihalli, Book – keeping and accounting, (R. Chand and Co. Delhi).
3. R. S. Singhal, Principles of Accountancy, (Nageen Prakashpvt. Lit. Meerut).
4. M. B. Kadkol, Book–Keeping and Accountancy,(Renuka Prakashan, Hubli)
5. Vithal, Sharma: Accounting for Management, Macmillan Publishers, Mumbai.
6. S. K. Bhattacharya and Jhon Dearden, Accounting for Management: Text and Cases, 3/e, Vikas publishing, 2018.

References:

1. B.S. Raman, Accountancy, (United Publishers, Mangalore).
2. Tulsian, Accounting and Financial Management – I: Financial Accounting – Person Education.

Title of Subject: DATA STRUCTURES USING 'C' (Major – 1)

COURSE CODE: 24MJBCA2L1	CIA Marks: 20	
SEMESTER: II	SEE Marks: 80	
Contact Hours: (L:T:P): 4-0-0	Credit: 04	Duration of Exam: 03

Course Outcomes: After completing this course satisfactorily, a student will be able to:

1. Describe how arrays, linked structures, stacks, queues, trees are represented in memory and used by algorithms
2. Describe common applications for arrays, linked structures, stacks, queues, trees
3. Write programs that use arrays, linked structures, stacks, queues, trees,
4. Compare alternative implementations of data structures with respect to performance
5. Describe the concept of recursion; give examples of its use.
6. Discuss the computational efficiency of the principal algorithms for sorting, searching.

UNIT – I: **12 Hours**

Introduction to Data structures: Definition; Types of data structures - Primitive & Non-primitive, Linear and Non-linear; Operations on data structures. Algorithm Specification, Performance Analysis, Performance Measurement

Recursion: Definition; Types of recursions; Recursion Technique Examples - Fibonacci numbers, GCD, Factorial, Comparison between iterative and recursive functions.

UNIT – II: **12 Hours**

Arrays: Basic Concepts, Definition, Declaration, Initialization, Operations on arrays; Types of arrays; Arrays as abstract data types (ADT); Representation of Linear Arrays in memory; Traversing line arrays; Inserting and deleting elements; **Sorting**: Selection sort, Bubble sort, Quick sort, Selection sort, Insertion sort; **Searching**: Sequential Search, Binary search; Iterative and Recursive searching; multidimensional Arrays, representation of multidimensional Arrays, Sparse matrices.

UNIT – III: **12 Hours**

Stacks: Basic Concepts, Definition and Representation of stacks; Operations on stacks; Applications of stacks; Infix, postfix and prefix notations; Conversion from infix to postfix using stack; Evaluation of postfix expression,**Tower of Hanoi.**

Queues: Basic Concepts, Definition and Representation of queues; Types of queues - Simple queues, Circular queues, Double ended queues, Priority queues; Operations on Simple queues;

UNIT – IV: **10 Hours**

Dynamic memory allocation: Static & Dynamic memory allocation; Memory allocation and de-allocation functions - malloc, calloc, realloc and free.

Linked list: Basic Concepts, Definition and Representation of linked list, Types of linked lists - Singly linked list, doubly liked list, Header liked list, Circular linked list; Representation of single Linked list in Memory; Operations on Singly linked lists – Traversing, Searching, Insertion, Deletion; Memory allocation; Garbage collection.

UNIT – V: **10 Hours**

Trees: Definition; Tree terminologies –node, root node, parent node, ancestors of a node, siblings, terminal & non-terminal nodes, degree of a node, level, edge, path, depth; **Binary tree:** Type of binary trees - strict binary tree, complete binary tree, binary search tree and heap tree; Array representation of binary tree. Traversal of binary tree; pre order, in order and Post order traversal; Reconstruction of a binary tree when any two of the traversals are given.

Text Books:

1. Kamthane: Introduction to Data Structure in C, Pearson Education 2005.
2. Langsam, Ausenstein Maoshe & M. Tanenbaum Aaron, Data Structure using C and C++ Pearson Education.

References Books:

1. Weiss: Data Structure and Algorithm Analysis in C, IInd Edition, Pearson Education.
2. Lipschutz: Schaum's outline series Data Structures, Tata McGraw Hill.
3. Tenenbaum: Data Structures using C, Pearson Education

Title of Subject: DATA STRUCTURES LAB' (Major – 1)		
COURSE CODE: 24MJBCA2P1	CIA Marks: 10	
SEMESTER: II	SEE Marks: 40	
Contact Hours: (L:T:P): 0-0-4	Credit: 02	Duration of Exam: 03

Part A

Algorithm Specification and Performance Analysis

- Write a program to implement and analyze the time complexity of a simple algorithm like Linear Search.
- Write a program to implement and analyze the space complexity of a recursive algorithm like calculating Fibonacci numbers.

Recursion Techniques

- Write a recursive function to compute the Fibonacci sequence up to n terms.
- Write a recursive function to find the GCD of two numbers.
- Write a recursive function to compute the factorial of a given number.
- Compare the performance of recursive and iterative versions of Fibonacci and factorial functions.

Basic Array Operations

- Write a program to perform insertion, deletion, and traversal of elements in a one-dimensional array.
- Write a program to implement and test various array operations (traversing, searching, insertion, deletion) on a two-dimensional array.

Sorting Algorithms

- Implement Selection Sort and analyze its time complexity.
- Implement Bubble Sort and analyze its time complexity.
- Implement Quick Sort and analyze its time complexity.
- Implement Insertion Sort and analyze its time complexity.

Searching Algorithms

- Implement Sequential Search and analyze its performance.
- Implement Binary Search and compare its performance with Sequential Search.

Sparse Matrices

- Write a program to represent and perform operations on a sparse matrix.

Part B

Stacks

- Implement a stack using arrays and perform operations like push, pop, and display.
- Write a program to convert an infix expression to a postfix expression using stack.
- Write a program to evaluate a postfix expression using stack.
- Implement the Tower of Hanoi problem using recursion.

Queues

- Implement a simple queue using arrays and perform operations like enqueue, dequeue, and display.
- Implement a circular queue using arrays and perform operations like enqueue, dequeue, and display.
- Implement a priority queue and perform necessary operations.
- Implement a double-ended queue (deque) and perform operations like insert at front, insert at rear, delete from front, and delete from rear.

Dynamic Memory Allocation

- Write a program to dynamically allocate memory for an array and perform operations on it using malloc, calloc, realloc, and free.

Linked Lists

- Implement a singly linked list and perform operations like insertion, deletion, and traversal.
- Implement a doubly linked list and perform operations like insertion, deletion, and traversal.
- Implement a circular linked list and perform operations like insertion, deletion, and traversal.
- Implement a header linked list and perform necessary operations.

Binary Trees

- Write a program to implement a binary tree and perform pre-order, in-order, and post-order traversals.
- Write a program to reconstruct a binary tree from given in-order and pre-order traversals.
- Implement a binary search tree (BST) and perform operations like insertion, deletion, and searching.

Heap Trees

- Implement a min-heap and max-heap and perform operations like insertion and deletion.

Evaluation Scheme for Lab Examination:

Assessment Criteria		**Marks**
Program – 1 from Part A	Writing the Program	07
	Execution	08
Program -2 from Part B	Writing the Program	07
	Execution	08
Practical Record		05
Viva-Voce		05
Total		**40**

Title of Subject: OOPs with C++ (Major – 2)

COURSE CODE: 24MJBCA2L2	**CIA Marks: 20**	
SEMESTER: II	**SEE Marks: 80**	
Contact Hours: (L:T:P): 4-0-0	**Credit: 04**	**Duration of Exam: 03**

Course Outcomes: After completing this course satisfactorily, a student will be able to:

1. Understanding about object-oriented programming and Gain knowledge about the capability to store information together in an object.
2. Understand about constructors and destructors which are special type of functions.
3. Understand Inheritance and operator overloading concepts and implement them
4. Use functions and pointers in your C++ program.
5. Write C++ programs using classes and objects.

UNIT – I: 12 Hours

Introduction: Object-Oriented Programming paradigm, basic concepts of object- oriented programming, benefits of OOPs, object-oriented languages, applications of OOP, structure of C++ program, creating the source file, compiling and linking.

C++ Basics: The iostream classes, C++ comments, C++ keywords, variable declaration, the const qualifier, the endl, setw, set Precision, Manipulators, the scope resolution operator, the new and delete operators, expressions and implicit conversions, operator precedence, control structures.

Functions: Function prototyping; call by reference, return by reference, inline functions, default argument, Const arguments, function overloading.

UNIT – II: 12 Hours

Classes and Objects: Specifying a class, defining member functions, making an outside function inline, nesting of member functions, private member function, arrays within a class, memory allocation for objects, static data member, static member functions, arrays of objects, object as function arguments. Constructors, parameterized constructors, multiple constructors with default arguments, dynamic initialization of objects, copy constructors, dynamic constructors, and destructors.

UNIT – III: 10 Hours

Operator Overloading: Fundamental of operator overloading, Restriction on operator overloading, Operator functions as a class member, Overloading unary operator: unary minus, increment operator Overloading binary operator: arithmetic operator, comparison operators, arithmetic assignment operator, Data conversion: conversion between basic to class types, conversion between objects and basic types, conversion between objects of different classes.

UNIT – IV: 10 Hours

Inheritance: Introduction to inheritance, Derived class & Base class: Specifying the derived class, access specifiers, accessing the base class members, the protected access specifier, derived class constructor, overriding member functions, types of inheritance: Single level inheritance, Multilevel inheritance, hybrid inheritance, multiple inheritance, public & private inheritance, member functions in multiple inheritance, constructors in multiple inheritance, Containership: classes within classes, Inheritance program development.

UNIT – V: 12 Hours

Virtual Functions: Normal member function accessed with pointers, virtual member function accessed with pointers, dynamic binding, pure virtual functions, Friend function: friends for functional notation, friend classes, this pointer, accessing member data with this, using this for returning values.

Templates & Exception Handling: Introduction, templates, class templates, function templates, member function templates, template arguments, Exception handling.

Text Book:

L E.Balaguruswamy: Object oriented Programming with C++ Tata McGraw Hill publications.

2. Lafore Robert: Object oriented Programming in Turbo C++ Galgotia Publications.

Reference:

1. Stanley B. Lippman, JoseeLajoie, Barbara E. Moo: C++ primer, 5th Edition, Addison-Wesley.
2. Prata : C++ primer Plus, 4th Edition. Person Education.
3. Strousstrup: The C++ programming Language Pearson Education.

Title of Subject: OOPs with C++ LAB (Major – 2)		
COURSE CODE: 24MJBCA2P2	CIA Marks: 10	
SEMESTER: II	SEE Marks: 40	
Contact Hours: (L:T:P): 0-0-4	Credit: 02	Duration of Exam: 03

Part A

Introduction to C++ and Basic Concepts

- Write a C++ program to display the structure of a basic C++ program with comments explaining each part.
- Write a C++ program that demonstrates the use of iostream classes and iomanip manipulators like endl, setw, and setprecision.

Control Structures

- Write a C++ program to demonstrate the use of various control structures (if, switch, for, while, do-while).

Functions

- Write a C++ program to demonstrate function prototyping and function overloading.
- Write a C++ program that uses call by reference and return by reference.
- Write a C++ program to demonstrate the use of inline functions.
- Write a C++ program that uses default and constant arguments in functions.

Basic Class Implementation

- Write a C++ program to define a class, its member functions, and demonstrate the creation of objects.
- Write a C++ program to demonstrate private member functions and inline member functions.
- Write a C++ program to show the use of static data members and static member functions.

Constructors and Destructors

- Write a C++ program to implement various types of constructors: default, parameterized, copy, and dynamic constructors.
- Write a C++ program to demonstrate the use of destructors.

Arrays and Objects

- Write a C++ program to demonstrate the use of arrays within a class and arrays of objects.

- Write a C++ program to pass objects as function arguments and return objects from functions.

Part B

Unary and Binary Operator Overloading

- Write a C++ program to overload the unary minus operator.
- Write a C++ program to overload the increment (++) and decrement (--) operators.
- Write a C++ program to overload arithmetic binary operators (e.g., +, -).
- Write a C++ program to overload comparison operators (e.g., ==, <, >).

Type Conversion

- Write a C++ program to demonstrate type conversion between basic types and user-defined types.
- Write a C++ program to demonstrate type conversion between objects of different classes.

Basic Inheritance Concepts

- Write a C++ program to implement single inheritance.
- Write a C++ program to implement multilevel inheritance.
- Write a C++ program to implement multiple inheritance.
- Write a C++ program to demonstrate the use of the protected access specifier and constructor in inheritance.

Advanced Inheritance

- Write a C++ program to implement hybrid inheritance.
- Write a C++ program to demonstrate public and private inheritance.
- Write a C++ program to show member functions in multiple inheritance.
- Write a C++ program to implement containership (classes within classes).

Virtual Functions

- Write a C++ program to demonstrate the use of normal member functions and virtual member functions accessed with pointers.
- Write a C++ program to implement dynamic binding using virtual functions.
- Write a C++ program to demonstrate pure virtual functions and abstract classes.

Friend Functions and This Pointer

- Write a C++ program to demonstrate friend functions and friend classes.
- Write a C++ program to demonstrate the use of the this pointer.

Templates and Exception Handling

- Write a C++ program to implement function templates.
- Write a C++ program to implement class templates.
- Write a C++ program to demonstrate template arguments.
- Write a C++ program to implement exception handling using try, catch, and throw.

Evaluation Scheme for Lab Examination:

Assessment Criteria		**Marks**
Program – 1 from Part A	Writing the Program	07
	Execution	08
Program -2 from Part B	Writing the Program	07
	Execution	08
Practical Record		05
Viva-Voce		05
Total		**40**

Title of Subject: DISCRETE MATHEMATICAL STRUCTURES (Major – 1)

COURSE CODE: 24MJBCA2L3	**CIA Marks: 20**	
SEMESTER: II	**SEE Marks: 80**	
Contact Hours: (L:T:P): 4-0-0	**Credit: 04**	**Duration of Exam: 03**

Course Outcomes:

1. To understand the basic concepts of Mathematical reasoning, sets and functions.
2. To understand various counting techniques and principle of inclusion and exclusions.
3. Understand the concepts of various types of relations, partial ordering and Equivalence relations.
4. Apply the concepts of generating functions to solve the recurrence relations.
5. Familiarize the fundamental concepts of graph theory and shortest path algorithm

UNIT – I: 12 Hours

The Foundations: Logic and proofs: Propositional Logic, Applications of Propositional Logic, Propositional Equivalences, Predicates and Quantifiers, Nested Quantifiers, Rules of Inference, Introduction to Proofs, Proof Methods and Strategy.

Basic Structures: Sets, Functions, Sequences, Sums, and Matrices: Sets, set operations, Functions, Sequences and Summations, matrices.

UNIT – II: 12 Hours

Counting: Basics of counting, Pigeonhole principle, Permutation and combination, Binomial Coefficient and Combination, Generating Permutation and Combination.

Advanced Counting Techniques: Applications of Recurrence Relations, Solving Linear Recurrence, Relations, Divide and Conquer Algorithms and Recurrence Relations, Generating functions, Inclusion-Exclusion,
Applications of Inclusion-exclusion.

UNIT – III: 12 Hours

Induction and Recursion: Mathematical Induction, Strong Induction and Well Ordering, Recursive Definitions and Structural Induction, Recursive Algorithms, Program Corrections.

Relation: Properties of relation, Composition of relation, Closer operation on relation, Equivalence relation and partition. Operation on relation, Representing relation.

UNIT – IV: 10 Hours

Graphs: Graphs and Graph models, Graph Terminology and Special Types of Graphs, Representing Graphs and Graph Isomorphism, Connectivity, Euler and Hamilton Paths, Shortest-Path Problems, Planar Graphs, Graph Coloring

UNIT – V: 10 Hours

Algorithmic approach on graph theory: Spanning trees, Connector problem, Minimal spanning tree, Fundamental circuits, connectedness and Components, The travelling salesman problem, shortest path from a specified vertex to another vertex-Dijkstra Algorithm.

<u>Text Books:</u>

1. Discrete Mathematics and Its Applications, Kenneth H. Rosen: Seventh Edition, 2012.
2. Discrete Mathematical Structure, Bernard Kolman, Robert C,Busby,Sharon Ross, 2003.
3. Graph Theory with Applications to Engg and Comp. Sci: Narsingh Deo-PHI 1986
4. Discrete and Combinatorial Mathematics Ralph P.Grimaldi, B.V.Ramatta, Pearson, Education, 5th Edition.
5. Discrete Mathematical Structures, Trembley and Manobar.

BCA/B.Sc Degree Examination,

SEP – QP - Pattern

Time: 3 Hours **Max. Marks: 80**

Section – A

Note: Answer all sub questions
Each question carries TWO mark. **(10 x 2 = 20)**

1.

a)
b)
c)
d)
e)
f)
g)
h)
i)
j)

Section – B

Note : Answer any Four questions
Each question carries FIVE marks. **(4 x 5 =20)**

2.
3.
4.
5.
6.
7.

Section – C

Note : Answer any Four questions
Each question carries TEN marks. **(4 x 10 =40)**

8.
9.
10.
11.
12.
13.

Note : 1. For Section –A , Two questions from each Unit.
2. For Section – B , One question from each Unit, and Q-7 must be from Unit 2 to 5.
3. For Section – C , One question from each Unit, and Q-13 must be from Unit 2 to 5.

BCA/B.Sc Degree Examination,

SEP – Scheme for Practical Examination

1. Writing Two Programs	: 14 Marks (for each 7 marks)
2. Execution of Two programs	: 16 Marks (for each 8 marks)
3. Practical record	: 05 Marks
4. Viva Voce	: 05 Marks
Total	: 40 Marks