SEMESTER-I

COURSE 2: PROBLEM SOLVING USING C

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. Understand the fundamentals of computer programming, Apply structured problem-solving approaches using algorithms, flowcharts, and C programming constructs.
2. Develop efficient logic using decision-making, loop, and jump control statements.
3. Utilize derived data types like arrays and strings for modular program design.
4. Design and implement modular solutions using functions, recursive logic, pointer operations, and dynamic memory management.
5. Handle complex data structures including structures, unions, and text file operations.

Course Outcomes:

At the end of the course, students will be able to:

1. Understand basic computing concepts, programming paradigms and write structured C programs.
2. Apply control flow statements to solve logical and repetitive tasks in C.
3. Implement arrays and string operations to manage and manipulate data efficiently.
4. Design modular code using functions, recursion, and appropriate parameter passing.
5. Utilize pointers and memory operations for effective data handling. Demonstrate competence in dynamic memory allocation and text file processing.

Unit 1. Introduction to computer programming:
Introduction, Types of software, Compiler and interpreter, Concepts of Machine level, Assembly level and high-level programming, Flowcharts and Algorithms, Fundamentals of C: History of C, Features of C, C Tokens-variables and keywords and identifiers, constants and Data types, Rules for constructing variable names, Operators, Structure of C program, Input /output statements in C-Formatted and Unformatted I/O

Unit 2. Control statements:
Decision making statements: if, if else, else if ladder, switch statements. Loop control statements: while loop, for loop and do-while loop. Jump Control statements: break,continue and goto.

Unit 3. Derived data types in C:
Arrays: One Dimensional arrays - Declaration, Initialization and Memory representation; Two Dimensional arrays -Declaration, Initialization and Memory representation. Strings: Declaring & Initializing string variables; String handling functions, Character handling functions

Unit 4. Functions:
Pointers: Pointer data type, Pointer declaration, initialization, accessing values using pointers. Pointer arithmetic, Pointers and arrays.

Function Prototype, definition and calling. Return statement. Nesting of functions. Categories of functions. Recursion (Basic Concept only). Parameter Passing by address & by value. Local and Global variables. Storage classes: automatic, external, static and register.

Unit 5. Dynamic Memory Management:
Introduction, Functions-malloc, calloc, realloc, free Structures: Basics of structure, structure members, accessing structure members, nested structures, array of structures, structure and functions, structures and pointers. Unions - Union definition; difference between Structures and Unions. Working with text files - modes: opening, reading, writing and closing text files.

Text Books:

1. Programming in ANSI C, E. Balagurusamy, Tata McGraw Hill, 6 th Edn,
2. Computer fundamentals and programming in C, Reema Theraja, Oxford University Press

Reference Books:

1. Let us C, Y Kanetkar, BPB publications
2. Head First C: A Brain-Friendly Guide, David Griffiths, Dawn Griffiths

Activities:

Outcome: Understand basic computing concepts, programming paradigms and write structured C programs.

Activity: Create a concept map of computing fundamentals and programming paradigms (procedural, structured, object-oriented). Then, they write a structured C program (e.g., a calculator or student grade system) using proper syntax, indentation, and modular design.

Evaluation Method: Rubric-based Code Review & Viva to check the

- The correctness of the concept map
- Correct use of structure (main + functions)
- Identification of paradigm used
- Code readability and documentation

Outcome: Apply control flow statements to solve logical and repetitive tasks in C.

Activity: Implement a program that solves a logic puzzle (e.g., number guessing game, pattern generation, or prime number finder) using if, switch, for, while, and do-while.

Evaluation Method: Automated Test Cases + Peer Review to check the

- Correct use of control statements
- Logical correctness of output

- Efficiency and edge case handling
- Peer feedback on clarity and logic

Outcome: Implement arrays and string operations to manage and manipulate data efficiently.

Activity: Build a program that stores and arranges student marks in ascending and descending order using arrays and performs string operations like concatenation, comparing, and formatting names.

Evaluation Method: Functional Demonstration + Code Walkthrough to check the

- Correct array and string usage
- Memory efficiency
- Handling of invalid inputs
- Explanation of sorting/searching logic

Activity:

- **Recursive Problem Solver**
  Students write a modular program to solve a recursive problem (e.g., factorial, Fibonacci, or Tower of Hanoi) using functions with parameters and return values.

Evaluation Method:

- **Code Trace + Written Quiz**
  - Correct function decomposition
  - Proper parameter passing (by value/reference)
  - Recursion depth and base case handling
  - Quiz on tracing recursive calls

Outcome: Utilize pointers and memory operations for effective data handling. Demonstrate competence in dynamic memory allocation and text file processing.

Activity: Create a program that dynamically stores user input (e.g., survey responses) using pointers and writes/reads the data to/from a text file.

Evaluation Method: Memory Debugging + File I/O Assessment to check the

- Proper use of malloc, calloc, realloc, and free
- Pointer arithmetic and dereferencing
- File creation, reading, writing, and error handling
- Use of tools like Valgrind or manual memory trace (Optional for Unix flavours)

SEMESTER-I

COURSE 2: PROBLEM SOLVING USING C

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

1. Write a program to check whether the given number is Armstrong or not.
2. Write a program to find the sum of individual digits of a positive integer.
3. Write a program to generate the first n terms of the Fibonacci sequence.
4. Write a program to find both the largest and smallest number in a list of integer values
5. Write a program to demonstrate change in parameter values while swapping two integer variables using Call by Value & Call by Address
6. Write a program to perform various string operations.
7. Write a program to search an element in a given list of values.
8. Write a program that uses functions to add two matrices.
9. Write a program to calculate factorial of given integer value using recursive functions
10. Write a program for multiplication of two N X N matrices.
11. Write a program to sort a given list of integers in ascending order.
12. Write a program to calculate the salaries of all employees using Employee (ID, Name, Designation, Basic Pay, DA, HRA, Gross Salary, Deduction, Net Salary) structure.
    a. DA is 30 % of Basic Pay
    b. HRA is 15% of Basic Pay
    c. Deduction is 10% of (Basic Pay + DA)
    d. Gross Salary = Basic Pay + DA+ HRA
    e. Net Salary = Gross Salary - Deduction
13. Write a program to read / write the data from / to a file.
14. Write a program to reverse the contents of a file and store in another file.
15. Write a program to create Book (ISBN,Title, Author, Price, Pages, Publisher)structure and store book details in a file and perform the following operations
    a. Add book details
    b. Search a book details for a given ISBN and display book details, if available
    c. Update a book details using ISBN
    d. Delete book details for a given ISBN and display list of remaining Books