Programming in C		Semester	I/II
Course Code	**1BEIT105/205**	CIE Marks	50
Teaching Hours/Week (L:T:P:S)	3:0:0:0	SEE Marks	50
Total Hours of Pedagogy	40	Total Marks	100
Credits	03	Exam Hours	3
Examination type (SEE)	Theory		

Course outcome (Course Skill Set)

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

CO1: Demonstrate fundamental concepts and language constructs of C programming.

CO2: Make use of control structures and arrays to solve basic computational problems.

CO3: Develop modular programs using user-defined functions for complex computational problems.

CO4: Construct user defined datatypes using structures, unions and enumerations to model simple real-world scenarios.

CO5: Choose suitable datatypes and language constructs to solve a given computational or real-world problem

Module-1

Introduction to Computing: Computer languages, Creating and Running Programs, System Development.

Overview of C: A Brief History of C, C Is a Middle-Level Language, C Is a Structured Language, C Is a Programmer's Language, Compilers Vs. Interpreters, The Form of a C Program, The Library and Linking, Separate Compilation, Compiling a C Program, C's Memory Map.

Expressions: The Basic Data Types, Modifying the Basic Types, Identifier Names, Variables, The Four C Scopes, Type Qualifiers, Storage Class Specifiers, Variable Initializations, Constants, Operators, Expressions.

Textbook 2: Chapter 1: 1.3, 1.4, 1.5; Textbook 1: Chapter 1, 2

Number of Hours: 08

Module-2

Console I/O: Reading and Writing Characters, Reading and Writing Strings, Formatted Console I/O, printf(), scanf().

Statements: True and False in C, Selection Statements, Iteration Statements, Jump Statements, Expression Statements, Block Statements.

Textbook 1: Chapter 8, 3

Number of Hours: 08

Module-3

Arrays and Strings: Single-Dimension Arrays, Generating a Pointer to an Array, Passing Single-Dimension Arrays to Functions, Strings, Two-Dimensional Arrays, Multidimensional Arrays, Array Initialization, Variable -Length Arrays.

Pointers: What Are Pointers?, Pointer Variables, The Pointer Operators, Pointer Expressions, Pointers and Arrays, Multiple Indirection, Initializing Pointers.

Textbook 1: Chapter 4, 5

Number of Hours: 08

Module-4

Functions: The General Form of a Function, Understanding the Scope of a Function, Function Arguments, argc and argv—Arguments to main(), The return Statement, What Does main() Return?, Recursion, Function Prototypes, Declaring Variable Length Parameter Declarations, The inline Keyword.

Pointers (Contd...): Pointers to Functions, C's Dynamic Allocation Functions.

Textbook 1: Chapter 5, Chapter 6

Number of Hours:08

Module-5

Structures, Unions, Enumerations, and typedef: Structures, Arrays of Structures, Passing Structure to Functions, Structure Pointers, Arrays and Structures within Structures, Unions, Bit-Fields, Enumerations, Using sizeof to Ensure Portability, typedef.

Textbook 1: Chapter 7

Number of Hours:08

Suggested Learning Resources:

Textbooks:

1. Schildt, Herbert. "C the complete reference", 4th Edition, Mc GrawHill.
2. Hassan Afyouni, Behrouz A. Forouzan. "A Structured Programming Approach in C", 4th Edition, Cengage.

Reference books:

1. Brian W. Kernighan and Dennis M. Ritchie, The 'C' Programming Language, 2nd Edition, Prentice Hall of India.
2. Reema Thareja, Programming in C, 3rd Edition, Oxford University Press, 2023.

Web links and Video Lectures (e-Resources):

1. elearning.vtu.ac.in/econtent/courses/video/BS/15PCD23.html
2. Introduction to Programming in C [https://onlinecourses.nptel.ac.in/noc23_cs02/preview]
3. C for Everyone: Programming Fundamentals [https://www.coursera.org/learn/c-for-everyone]
4. Computer Programming Virtual Lab [https://cse02-iiith.vlabs.ac.in/exp/pointers/]
5. C Programming: The ultimate way to learn the fundamentals of the C language [https://www.pdfdrive.com/c-programming-the-ultimate-way-to-learn-the-fundamentals-of-the-c-language-e187584209.html]
6. C Programming: The Complete Reference [https://viden.io/knowledge/programming-in-c-language/attachment/28313/c-the-complete-reference-herbert-schildt-4th-edition-pdf/preview]
7. https://infyspringboard.onwingspan.com/web/en/app/toc/lex_auth_01384323703937433634517_shared/overview
8. C programming Tutorial: https://www.geeksforgeeks.org/c/c-programming-language/.

Teaching-Learning Process (Innovative Delivery Methods):

The following are sample strategies that educators may adopt to enhance the effectiveness of the teaching-learning process and facilitate the achievement of course outcomes.

1. Flipped Classroom
2. Problem-Based Learning (PBL)
3. Case-Based Teaching
4. Simulation and Virtual Labs
5. ICT-Enabled Teaching

Assessment Structure:

The assessment in each course is divided equally between Continuous Internal Evaluation (CIE) and the Semester End Examination (SEE), with each carrying 50% weightage.

- To qualify and become eligible to appear for SEE, in the **CIE**, a student must score at least **40% of 50 marks**, i.e., **20 marks**.
- To pass the **SEE**, a student must score at least **35% of 50 marks**, i.e., **18 marks**.
- Notwithstanding the above, a student is considered to have **passed the course,** provided the combined total of **CIE and SEE is at least 40 out of 100 marks**.

Continuous Comprehensive Assessments (CCA):

CCA will be conducted for a total of 25 marks. It is recommended to include a maximum of two learning activities aimed at enhancing the holistic development of students. These activities should align with course outcomes and promote higher-order thinking and application-based learning.

Learning Activity -1: Programming Assignment (Marks- 25)

INSTRUCTIONS:

1. Course instructor will refer to HackerRank/HackerEarth/LeetCode or any other platform to derive the questions for problem-solving.
2. Course Instructor must identify programming problems from these sections: Statements (control), Arrays, Strings, Structures & Unions and Functions.
3. Courser instructor will assign THREE questions from each section to the students for design of algorithm, program and coding/execution.
4. Students must demonstrate the solutions to the course instructor and submit the record containing algorithm, program, debugging/execution and results with observations.
5. Course instructor must evaluate the student performance as per the rubrics.

Rubrics for Learning Activity-1 (Programming Assignment):

Component & CO-PO Mapping	**Outstanding (5)**	**Exceeds Expectations (4)**	**Meets Expectations (3)**	**Needs Improvement (2)**	**Unsatisfactory (1)**
Clarity & Simplicity of algorithm/program [CO1] [PO9]	Algorithm/Programs are self-explanatory, specific, and well-structured for the intended activity; no ambiguity is present.	Programs are clear and mostly specific; minor ambiguity is present.	Programs are somewhat clear but could be more specific; moderate ambiguity.	Programs are vague and lack clarity; high ambiguity.	Programs are unclear, incomplete, or irrelevant to the activity.
Appropriate Use of language constructs and design of algorithm/program [CO4, CO5] [PO1, PO3]	Demonstrates precise and creative usage of the language construct and structured programming	Correctly applies the language construct with minor gaps or missed opportunities.	Uses the language construct, but with partial understanding or inconsistent usage.	Limited understanding of the language construct; incorrect or weak usage.	No evidence of correct/relevant language construct use.
Compilation, Debugging, Analysis & Comparison of Results for various cases. [CO2, C03] [PO2, PO4, PO5]	Provides clear and correct results with analysis for multiple cases; comparisons among cases highlight key strengths and weaknesses.	Provides correct results with analysis for multiple cases, though slightly less detailed.	Provides correct results with limited analysis; comparisons are present but shallow.	Provides correct results. Minimal analysis: comparisons are weak or incomplete.	Results are partially correct. No meaningful analysis or comparison.
Creativity, efficiency of Problem-Solving/program [CO2, CO3] [PO3, PO11]	Demonstrates outstanding creativity and innovation in writing programs, especially for problem-solving or design tasks.	Demonstrates creativity and some innovation; Program solutions are practical.	Shows moderate creativity; programs are functional but not innovative.	Minimal creativity; programs are repetitive or unimaginative.	No creativity or problem-solving/Programming is evident.
Documentation & Reflection [CO1, CO4, CO5] [PO8/PO9/PO11]	Documentation is complete, well-organized, and includes deep reflection on improvements across iterations.	Documentation is complete with some reflection on program refinement.	Documentation is present but lacks detail or depth in reflection.	Incomplete documentation; reflection is minimal.	No documentation or reflection provided as per schedule.