INTRODUCTION TO C PROGRAMMING		Semester	I/II
Course Code	**1BPLC205E/105E**	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	3:02:0	SEE Marks	50
Total Hours of Pedagogy (Theory and Lab hours)	40 + 24 (Practical)	Total Marks	100
Credits	4	Exam Hours	3
Examination type (SEE)	Theory		

Course outcomes (Course Skill Set)

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

CO1: Explain the fundamental structure of a C program and primitive constructs.
CO2: Apply decision-making and iterative control structures to solve simple computational problems.
CO3: Develop programs using arrays and string operations to solve real-world problems.
CO4: Construct user-defined functions to modularize the solution to the given problems.
CO5: Build programs using structures and pointers for complex data representation and access.

Module-1

Flowchart and Algorithms: Art of Programming through Algorithms & Flowcharts.
Overview of C: History of C, Importance of C, Basic Structure of C Programs, Programming Style, Compiling and Executing a 'C' Program.
Constants, Variables and Data Types: Character Set, C Tokens, Keywords and Identifiers, Constants, Variables, Data Types, Declaration of Variables, Assigning Values to Variables, Defining Symbolic Constants, Declaring a Variables as Constants and Volatile, Input/Output Statements in C.

Textbook: Chapter 1. 6, 2.1, 2.2, 2.8, 2.9, 2.10, Chapter 3.2 to 3.14, Chapter 5.1 to 5.5

Number of Hours: 8

Module-2

Operators: Introduction to Operators, Arithmetic Operators, Relational Operators, Logical Operators, Assignment Operators, Increment and Decrement Operators, Conditional Operators, Precedence of Arithmetic Operators.
Decision Making, Branching, Looping: Introduction, Decision Making with IF Statement, Simple IF Statement, The IF..ELSE Statement, Nesting of IF..ELSE Statements, The ELSE IF Ladder, The Switch Statement, The ?: Operator, The GOTO Statement, WHILE, DO, FOR, Jumps in LOOPS.

Textbook: Chapter 4.1 to 4.7, 4.12, Chapter 6.1 to 6.9, Chapter 7.1 to 7.5 Number of Hours: 8

Module-3

Arrays and Strings: Introduction, Declaration and Initialization of One-dimensional and Two-Dimensional Arrays, Declaring and Initializing String Variables, Example programs using arrays ,Reading Strings from Terminal, Writing Strings to Screen, Arithmetic Operations on Characters, Comparison of Two Strings, String-handling Functions.

Textbook: Chapter 8.1 to 8.6, Chapter 9.2 to 9.5, 9.7, 9.8 Number of Hours: 8

Module-4

User-defined Functions: Introduction, Need for User-defined Functions, A Multi-functional Program, Elements of User-defined Functions, Definition of Function, Return Values and their Types, Function Calls, Function Declaration, No Arguments and no Return Values, Arguments but no Return Values, Nesting of Functions.

Textbook: Chapter 10.1 to 10.8, 10.10 to 10.14

Number of Hours:8

Module-5

Structures and Pointers: Introduction, Defining a Structure, Declaring and Accessing Structure Variables and Members, Structure Initialization, Copying and Comparing Structure Variables, Array of Structures, Arrays within Structures.
Pointers: Introduction, Understanding Pointers, Accessing the Address of Variable, Declaring pointer variables, initialization of pointers, accessing variables through its pointer.

Textbook: Chapter 11.1 to 11.6, 11.8, 11.19, Chapter 12.1 to 12.6

Number of Hours:8

PRACTICAL COMPONENT OF IPCC

1. Develop a program to calculate the temperature converter from degree to Fahrenheit.
2. Develop a program to find the roots of quadratic equations.
3. Develop a program to find whether a given number is prime or not.
4. Develop a program to find key elements in an array using linear search.
5. Given age and gender of a person, develop a program to categorise senior citizen (male & female).
6. Generate Floyd's triangle for given rows.
7. Develop a program to find the transpose of a matrix.
8. Develop a program to concatenate two strings, find length of a string and copy one string to other using string operations.
9. Develop a modular program to find GCD and LCM of given numbers.
10. Develop a program to declare the structure of employees and display the employee records with higher salary among two employees.
11. Develop a program to add two numbers using the pointers to the variables**.**
12. Develop a program to find the sum of digits of a given number.
13. Develop a program to perform Matrix Multiplication.
14. Develop a program to create an array of structures to store book details and check whether a specific book, as requested by the user, is available or not.

Suggested Learning Resources: (Textbook/ Reference Book/ Manuals):

Textbooks:

1. Programming in ANSI C, 9e, E Balaguruswamy, Tata McGraw Hill Education.

Reference books / Manuals:

1. PROGRAMMING IN C, Reema Thareja, Oxford University, Third Edition, 2023**.**
2. The 'C' Programming Language, Brian W. Kernighan and Dennis M. Ritchie, Second Edition, Prentice Hall of India, 2015

Web links and Video Lectures (e-Resources):

1. elearning.vtu.ac.in/econtent/courses/video/BS/15PCD23.html

2. https://nptel.ac.in/courses/106/105/106105171/ MOOC

Courses can be adopted for more clarity in understanding the topics and verities of problem-solving methods.

- https://www.tutorialspoint.com/what-is-an-algorithm-and-flowchart-in-c-language
- https://www.tutorialspoint.com/cprogramming/c_data_types.htm
- https://www.tutorialspoint.com/cprogramming/c_operators.htm
- https://www.ccbp.in/blog/articles/decision-making-statements-in-c
- https://www.tutorialspoint.com/cprogramming/c_arrays.htm
- https://www.geeksforgeeks.org/variables-in-c/
- https://www.w3schools.com/c/c_arrays.php
- https://www.programiz.com/c-programming/c-strings
- https://www.programiz.com/c-programming/c-pointers
- https://www.scaler.com/topics/c/structures-c/

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 for each course is equally divided between Continuous Internal Evaluation (CIE) and the Semester End Examination (SEE), with each component carrying **50% weightage** (i.e., 50 marks each).
The CIE Theory component will be 30marks and CIE Practical component will be 20 marks.

The CIE Theory component consists of IA tests for 25 marks and Continuous Comprehensive Assessments (CCA) for 5 marks. The CIE Practical component for continuous assessments will be for 15 marks through rubrics and for lab tests will be for 5 marks.

- To qualify and become eligible to appear for SEE, in the **CIE theory component**, a student must score at least **40% of 30 marks**, i.e., **12 marks**.
- To qualify and become eligible to appear for SEE, in the **CIE Practical component**, a student must secure **a minimum of 40% of 20 marks**, i.e., **08 marks**.
- To pass the **SEE**, a student must secure **a minimum of 35% of 50 marks**, i.e., **18 marks**.
- A student is deemed to have **successfully completed the course** if 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 5 marks. It is recommended to include any, one learning activity aimed at enhancing the holistic development of students. This activity should align with course outcomes and promote higher-order thinking and application-based learning.

Learning Activity -1: Programming Assignment (Marks- 5)

INSTRUCTIONS:

1. Course instructor will refer to HackerRank 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 question ONE from each section to the students for design of algorithm/flowchart, program and coding/execution.
4. Students must demonstrate the solutions to the course instructor and submit the record containing algorithm/flowchart, program, debugging/execution and results with observations.
5. Course instructor must evaluate the student performance as per the rubrics.

Rubrics for Learning Activity (Based on the nature of learning activity, Develop the rubrics for each activity):

Note: Marks obtained (25) is scaled down to 5.

Rubrics for Learning Activity:

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 [CO2-5] [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. [CO2-5] [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	Demonstrates outstanding	Demonstrates creativity and	Shows moderate creativity;	Minimal creativity:	No creativity or problem-

Problem-Solving/program [CO2-5] [PO3, PO11]	creativity and innovation in writing programs, especially for problem-solving or design tasks.	some innovation; Program solutions are practical.	programs are functional but not innovative.	programs are repetitive or unimaginative.	solving/Programming is evident.
Documentation & Reflection [CO1-5] [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.

Rubrics for CIE – Continuous assessment:

Component & CO-PO Mapping	Outstanding (5)	Exceeds Expectations (4)	Meets Expectations (3)	Needs Improvement (2)	Unsatisfactory (1)
Fundamental Knowledge: Understanding the problem statement [CO1] [PO1, PO2]	The student has in depth knowledge of the topics related to the problem. Student is able to completely understand the problem definition.	Student has good knowledge of some of the topics related to problem. Student is able to understand the problem definition.	Student is capable of narrating the answer but not capable to show in depth knowledge and the problem definition.	Student has not understood the concepts partially. Student is able to partially understand the problem definition	Student has not understood the concepts and the problem definition clearly.
Design of algorithm/flow chart and program [CO2-5] [PO2, PO3]	Student is capable of discussing more than one design for his/her problem statement and capable of proving the best suitable design with proper reason.	Student is capable of discussing few designs for his/her problem statement but not capable of selecting best.	Student is capable of discussing single design with its merits and de-merits.	Student is capable of explaining the design.	Student is capable of explaining the design partially.
Implementation (Program coding) with suitable tools [CO2-5] [PO5, PO8]	Student is capable of implementing the design with best suitable language structure considering optimal solution/optimal efficiency.	Student is capable of implementing the design with best suitable language structure and should be capable of explaining it.	Student is capable of implementing the design with proper explanation.	Student is capable of implementing the design.	Student is capable of implementing the design with errors.

Program debugging and testing with suitable tools [CO2-5] [PO5, PO8]	Student is capable to compile and debug the program with no errors (syntax, semantic and logical).	Student is able to compile and debug the program with errors (syntax, semantic and logical) and rectified errors with full understanding of error descriptions.	Student is able to compile and debug the program with errors (syntax, semantic and logical) and rectified errors with partial understanding of error descriptions.	Student is able to compile and debug the program with errors (syntax, semantic and logical) and rectified errors with no understanding of error descriptions.	Student is able to compile and debug the program with errors (syntax, semantic and logical) and rectified errors with assistance.
Results & interpretation /analysis [CO1-5] [PO4]	Student is able to run the program on various cases and compare the result with proper analysis.	Student is able to run the program for all the cases.	Student is able to run the code for few cases and analyze the result.	Student is able to run the program but not able to analyze the result.	Student is able to run the program but not able to verify the correctness of the result.
Demonstration and documentation [CO1-5] [PO8, PO9, PO11]	Demonstration and lab record is well-organized, with clear sections. The record is well structured with suitable formatting (e.g: font, spacing, labelling of figures and tables, equations numbered and etc).	Demonstration and lab record is organized, with clear sections, but some sections are not well-defined. The record is structured with formatting (e.g: font, spacing, labelling of figures and tables, equations numbered and etc).	Demonstration and lab record lacks clear organization or structure. Some sections are unclear or incomplete. The record is partially structured with formatting (e.g: font, spacing, labelling of figures and tables, equations numbered and etc).	Demonstration and lab record is poorly organized, with missing or unclear sections. The record is not properly structured with suitable formatting (e.g: font, spacing, labelling of figures and tables, equations numbered and etc).	Demonstration and lab record is poorly organized, with missing sections. Record not submitted on time. The record is not structured with minimum formatting (e.g: font, spacing, labelling of figures and tables, equations numbered and etc).

Rubrics for CIE Test:

Component & CO-PO Mapping	Excellent (5)	Good (4)	Fair (3)	Marginal (2)	Unsatisfactory (1)
Fundamental Knowledge (2) [CO1] [PO1]	The student has well depth knowledge of the topics related to the problem & course	Student has good knowledge of some of the topics related to problem & course	Student has average knowledge of some of the topics related to problem & course	Student is capable of narrating the answer but not capable to show in depth knowledge	Student has not understood the concepts clearly
Understanding of problem definition (1) [CO2+-5] [PO2]	Student is able to completely understand the problem definition	Student is able to understand the problem definition but not clearly	Student has a basic understanding of the problem definition that is partial or superficial	Student is able to Shows minimal or unclear understanding of the problem definition	Student is not able to understand the problem definition
Design and Implementation (3) [CO2-5] [PO3]	Student is capable of design and implementing with best suitable construct for the given problem definition	Student is capable of design and implementing with some construct for the given problem definition	Student is capable of design and implementing the core part of the construct for the given problem definition	Student is partially capable of design and implementing with some algorithm for the given problem definition	Student is not capable of design and implementing
Result & Analysis (2) [CO2-5] [PO4]	Student is able to run the program on various data inputs and compare the result with proper inference.	Student will be able to run the program on various data inputs and fair knowledge in comparing the result with proper inference	Student will be able to run the code for few data/datasets and analyze the output.	Student will be able to run the code for few data inputs but not analyze the output.	Student will be not able to run the program and not able to analyze the result.
Communication (Viva voce) (2) [CO1-5] [PO8, PO9]	Good Verbal & nonverbal communication skills with precise and correct terminologies/ answers.	Good verbal Communication skills with precise and correct terminologies/ answers.	Average Communication but with precise and correct terminologies/ answers.	Average Communication but with imprecise and incorrect terminologies/ answers	Poor Communication (Minimal interaction/answers)