GANPAT UNIVERSITY

FACULTY OF ENGINEERING & TECHNOLOGY

Programme	Bachelor of Technology	Branch/Spec.	All
Semester	I	Version	1.0.0.0
Effective from Academic Year	2026-27	Effective from the batch admitted in	July 2026
Course Code	2ESC1107	Course Name	Programming for Problem Solving
Course Category	Engineering Science Courses (ESC)		

Teaching Scheme						Examination scheme (Marks)			
(Per week)	Lecture (DT)		Practical (Lab.)		Total		CE	SEE	Total
	L	TU	P	TW					
Credit	3	0	1	0	4	Theory	50	50	100
Hours	3	0	2	0	5	Practical	25	25	50

Pre-requisites:

Basic understanding of computer fundamentals and foundational mathematics

Course Outcomes

COs	Description
CO1	Apply algorithms and flowcharts to solve basic mathematical and engineering problems.
CO2	Develop, compile, debug, and execute C programs on Linux and Windows platforms.
CO3	Use appropriate data types, control structures, arrays, pointers, and files in C programming.
CO4	Design structured programs and demonstrate self-learning and lifelong learning skills.

Theory Syllabus

Unit	Content	Hours
1	**Computational thinking foundations** **Introduction to Computational Thinking:** What is Computational Thinking? Why is it important for everyone (not just programmers)? The four pillars: Decomposition (breaking problems into smaller parts), Pattern Recognition (finding similarities), Abstraction (focusing on important details), Algorithm Design (step-by-step solutions). **Problem Decomposition & Pattern Recognition:** Breaking complex problems into manageable sub-problems. Identifying patterns in data and problems. Real-world examples: Planning a trip, organizing a birthday party, solving a puzzle. **Algorithms & Flowcharts:** What is an algorithm? Characteristics of good algorithms. Flowchart symbols (Start/End, Process, Decision, Input/Output, Connector). Converting real-world problems to flowcharts. Pseudocode basics.	08
2	**Introduction to C programming** **Computer Fundamentals & Programming Basics:** Components of a computer system (CPU, Memory, Storage - simple explanation with analogies). What is a program? Machine language vs High-level language. Compilation:Source code, Object code, Executable. History of C language. Why learn C? (Foundation for other languages, system programming, embedded systems). **First C Program & Basic I/O:** Structure of a C program (#include, main function, return statement). "Hello World" program - detailed explanation of each line. printf() function - displaying output. Escape sequences (\n, \t). Common beginner errors: missing semicolons, wrong brackets, case sensitivity. **Variables, Data Types & Input:** What is a variable? (Box with a name analogy).	08

	Variable naming rules. Data types: int (whole numbers), float (decimals), char (single character). Why different data types? Memory and precision. Declaration and initialization. scanf() function - taking user input. Format specifiers (%d, %f, %c).	
3	**Operators & Decision making** **Operators & Expressions:** Arithmetic operators (+, -, *, /, %). Integer division vs float division. Modulus operator applications (odd/even, digit extraction). Assignment operators (=, +=, -=). Increment/Decrement (++, --). Operator precedence (BODMAS in programming). Type conversion (implicit and explicit casting). **Decision Making - if, if-else:** Relational operators (>, <, >=, <=, ==, !=). Logical operators (&&, ||, !). Simple if statement. if-else statement. Nested if-else. Common mistakes: = vs ==, missing braces. **Multiple Choices - else-if & switch:** else-if ladder for multiple conditions. switch-case statement. break and default in switch. When to use if-else vs switch. Menu-driven program design**.**	08
4	**Loops & Iteration** **Introduction to Loops - while & do-while:** Why loops? (Printing 1-100 without 100 printf statements). while loop syntax and working. Loop control variable. Infinite loops and how to avoid them. do-while loop - guaranteed first execution. Difference between while and do-while **for Loop & Nested Loops:** for loop syntax - initialization, condition, update in one line. Choosing between for, while, do-while. Nested loops - loop inside a loop. How nested loops work (outer loop iteration = full inner loop execution). **Loop Control & Pattern Printing:** break statement - exit loop early. continue statement - skip current iteration. Practical applications of break and continue. Pattern printing with nested loops (triangles, pyramids, diamonds). Logic building for patterns.	08
5	**Arrays & Strings** **One-Dimensional Arrays:** What is an array? (Collection of similar items, like a row of mailboxes). Array declaration and initialization. Accessing elements using index (0-based indexing). Reading and printing array elements. Array bounds and common errors. **Array Operations & Searching:** Finding sum, average, maximum, minimum of array elements. Linear search algorithm - finding an element. Counting occurrences. Reversing an array. Basic sorting concept (Bubble sort introduction). **Strings (Character Arrays):** Strings as character arrays with null terminator (\0). String declaration and initialization. Reading strings (scanf, gets, fgets). Printing strings. String length (manual counting, then strlen). String copy and compare concepts. Basic string functions from string.h (strlen, strcpy, strcmp, strcat).	08
6	**Functions & Modular programming** **Introduction to Functions:** Why functions? (Reusability, organization, debugging). Function declaration (prototype). Function definition. Function call. void functions (no return value). Functions with return values. **Parameters & Arguments:** Formal parameters vs Actual arguments. Call by value mechanism. Scope of variables - local vs global. Passing multiple parameters. Returning values from functions. **Pointers:** Idea of pointers, Defining pointers, Use of Pointers in self-referential structures, Pointers and Function Arguments, Pointers and Arrays, Address Arithmetic, character Pointers and Functions, Pointer Arrays, Pointer to functions.	05

Practical and Self Learning Content	
Practical, assignments, quiz, industrial visit, field survey and tutorials are based on the above syllabus.	
Text Books	
1	“Programming in ANSI C” by E Balagurusami, Tata MacGraw-Hill
Reference Books	
1	“Let’s C” by YashvantKanetkar, BPB Publication
2	“Programming in C” by Ashok Kamthane, Pearson Publication
3	“The C Programming Language” by Brian W. Kernighan and Dennis Ritchie, Prentice-Hall
4	“Computer Programming in C” by V Rajaraman, PHI
5	“Outline of Programming with C” by Byron Gottfried and Schaum’s , McGraw-Hill
ICT/MOOCs Reference	
1	https://nptel.ac.in/courses/106104128
2	https://www.mooc-list.com/course/c-everyone-structured-programming-coursera

Mapping of COs, POs, and PSOs														
COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
CO1	3	3	2	1	2	0	0	1	1	0	2	3	1	2
CO2	2	2	3	2	3	0	0	2	1	0	2	3	3	2
CO3	3	2	3	1	3	0	0	1	1	0	2	3	3	2
CO4	2	3	3	2	2	1	1	2	2	1	3	3	2	3

Bloom's Taxonomy Level				
Unit	**Unit Title**	**Aligned COs**	**Learning Hours**	**BTL Level**
1	Computational thinking foundations	CO1, CO4	08	R,U
2	Introduction to C programming	CO2, CO3	08	U
3	Operators & Decision making	CO2, CO3	08	A
4	Loops & Iteration	CO2, CO3, CO4	08	N
5	Arrays & Strings	CO3, CO4	08	N
6	Functions & Modular programming	CO3, CO4	05	N

Note:

- Version 1.0.0.0 (First Digit= New syllabus/Revision in Full Syllabus, Second Digit=Revision in Teaching Scheme, Third Digit=Revision in Exam Scheme, Forth Digit= Content Revision)
- 1 Hour Lecture = 1 Credit, 1 Hour Tutorial = 1 Credit, 2 Hours Practical = 1 Credit, 2 Hours Internship/Project/Seminar = 1 Credit
- Bloom's Taxonomy Level (BTL): R: Remember, U: Understand, A: Apply, N: Analyze, E: Evaluate, and C: Create