Course Title: Programming Fundamentals and 'C' Programming
Course No.: CSIT.115
Nature of the Course: Theory+Lab
Level: B.Sc. CSIT
Year: First
Semester: First

Credit: 3
Number of hours per week: 3
Total hours: 48

1. Course Introduction

The course intends to enable the students to be acquainted with the basic concepts of programming methodology, 'C' Programming language.

2. Objectives

At the end of this course the students should be able:

- To develop a programming logic.
- To teach basic principles of programming.
- To develop skills for writing programs using 'C'.

3. Specific Objectives and Contents

Specific Objectives	Contents
• Define algorithm, use of algorithms • Describe different notations of algorithms • State standard notations and common functions • Classify different Pseudo-code Conventions • Develop fundamental algorithms • Write different algorithms for different problems • Differentiate different programming approaches and their benefits. • Understand the basic structure of C program • Understand different types of data types and qualifiers in terms of memory requirement and range. • Write various programs using different data types, qualifiers.	**Unit I: Introduction To Algorithms and C (8 Hrs)** **Fundamentals of algorithms:** Notion of an algorithm. Pseudo-code conventions like assignment statements and basic control structures. **Algorithmic problems** : Develop fundamental algorithms for (i)Exchange the values of two variables with and without temporary variable, (ii) Counting positive numbers from a set of integers, (iii) Summation of set of numbers, (iv) Reversing the digits of an integer, (v) Find smallest positive divisor of an integer other then 1, (vi) Find G.C.D. and L.C.M. of two as well as three positive integers (vii) Generating prime numbers. **Different approaches in programming:** Procedural approach, Object Oriented approach, Event Driven approach. **Structure of C:** Header and body, Use of comments, Compilation of program. **Data Concepts:** Variables, Constants, data types like: int, float char, double and void. Qualifiers: Short and ling size qualifiers, signed and unsigned qualifiers. Declaring variables. Scope of the variables according to block. Hierarchy of data types.
• Write various 'C' programs to perform various types of operations on the data values which are to be processed. • Input various types of data and obtain the output in a desired form • Alter the sequence of the execution of the program • Set up loops to repeat a set of	**Unit II : Basic of C (4 Hrs)** **Types of operators:** Arithmetic, Relational, Logical, Compound Assignment, Increment and decrement, Conditional or ternary, Bitwise and Comma operators, Precedence and order of evaluation. Statements and Expressions. **Type Conversions :** Automatic and Explicit type conversion **Data Input and Output function :** Formatted I/O: printf(), scanf(), Character I/O format : getch(), gerche(), getchar(), getc(), gets(),

statements, desired number of times • transfer control to different statements in the program	putchar(), putc(), puts() **Iterations:** Control statements for decision making: (i) Branching: if statement, else.. If statement, switch statement (ii) Looping: while loop, do… while, for loop. (iii) Jump statements: break, continue and goto.
• Understand what arrays are • What is the need for arrays • How arrays can be used in C Language • Declare and use one dimensional and two dimensional arrays • Understand the need for character and string variables • Declare and use character and string variables • Use functions to handle character and string data • Understand the Purpose of Sorting • Understand the different methods of Sorting. • Identify the advantages of different algorithms of Sorting • Be able to write programs in C to implement the algorithms for Sorting • Explain what is meant by Efficiency of an algorithm • Compare algorithms for Efficiency	**Unit III : Arrays, Strings and Sorting Techniques (8 Hrs)** **Arrays :** (One and multidimensional), declaring array variables, initialization of arrays, accessing array elements. **Strings:** Declaring and initializing String variables. Character and string handling functions. **Sorting Algorithms :** Bubble, Selection, Insertion and Merge sort, Efficiency of algorithms, Implement using C.
• Understand what Functions are and why are they needed. • Be able to define a Function in terms of its arguments and return values • Understand when and how to use Functions • Understand what are Macros and why they are needed • Explain how Macros are different from functions? • Understand what is Recursion? • Explain the Advantages of Recursion • Write programs for some standard situations for recursive functions such as Fibonacci Sequence and Towers of Hanoi • Be able to understand situations where recursion is needed • Understand the concept of a storage class • Understand the different storage classes • Understand the concept of scope,	**Unit IV: Functions, Storage Classes and Recursion (8 Hrs)** **Functions:** Global and local variables, Function definition, return statement, Calling a function by value, Macros in C, Different between functions and macros. **Storage classes :** Automatic variables, External variables, Static variables, Register variables. **Recursion:** Definition, Recursion function algorithms for factorial, Fibonacci sequence, Tower of Hanoi. Implement using C

visibility and longevity of a variable • Understand which storage class should be used under what circumstances • Learn the advantages and disadvantages of each storage class	
• Understand what are structures and why they are needed • Be able to define a structure • Be able to read and assign values to elements in a structure • Be able to understand the relationship between arrays and structures • Be able to define structures within structures • Be able to understand the relationship between structures and functions • Be able to understand what are unions • Write programs involving the use of structures	**Unit V: Structure and Union (4 Hrs)** **Structure:** Declaration of structure, reading and assignment of structure variables, Array of structures, arrays within structures, within structures, structures and functions. **Unions :** Defining and working with union
• Understand the pointers • Write dynamic programs • Understand strength of pointers • Store data in files • Read data from files • Understand File Handling Functions	**Unit VI: Pointers and File Handling (6 Hrs) Pointer:** Fundamentals, Pointer variables, Referencing and dereferencing, Pointer Arithmetic, Chain of pointers, Pointers and Arrays, Pointers and Strings, Array of Pointers, Pointers as function arguments, Functions returning pointers, Pointer to function, Pointer to structure, Pointers within structure. **File Handling:** Different types of files like text and binary, Different types of functions fopen(), fclose(), fputc(), fscanf(), fprintf(), getw(), putw(), fread(), fwrite(), fseek() **Dynamic Memory Allocation:** malloc(), calloc(), realloc(), free() and size of operator.
• Define a Linear Link List and list its features. • Understand the advantages & shortcomings of link list over an array. • Differentiate between Link List & Array. • Write & Explain the basic operations of Linear Link List. • Understand how to implement a link list. • Write a program in C to implement linear link list.	**Unit VII : Link Lists (4 Hrs)** **Linear Link lists:** Representation of link list in memory, Algorithms for traversing a link list, searching a particular node in link list, insertion into link list (insertion at the beginning of a node, insertion after a given node) deletion from a link list. Implement using C.

• Define a stack and its features. • Write Algorithms for the basic operations of Stack. • Understand the difference between Stack & Array. • Understand how an Array is used to implement a Stack. • Write a program in C to implement Stack.	**Unit VIII: Stacks** (3 Hrs) **Stacks:** Definition, Array representation of stacks, Algorithms for basic operators to add and delete an element from the stack, Implement using C.
• Define a queue and state its features. • State the applications that use queues. • State the basic operations of a queue. • Differentiate between straight queue and circular queue. • Implement queues using arrays and linked lists.	**Unit VIII: Queues** (3 Hrs) **Queues:** Representation of queue, Algorithm for insertion and deletion of an element in a queue, Implement using C.

Evaluation System:

Undergraduate Programs				
External Evaluation	**Marks**	**Internal Evaluation**	**Weightage**	**Marks**
End semester examination	60	Assignments	10%	40
(Details are given in the separate table at the end)		Quizzes	10%	
		Attendance	10%	
		Presentation	10%	
		Term papers	10%	
		Mid-Term exam	40%	
		Group work	10%	
Total External	60	Total Internal	100%	40
Full Marks 60+40 = 100				

External evaluation:
End semester examination: It is a written examination at the end of the semester. The questions will be asked covering all the units of the course. The question model, full marks, time and others will be as per the following grid.

Full Marks: 100, Pass Marks: 50, Time: 3 Hrs

Nature of question	Total questions to be asked	Total questions to be answered	Total marks	Weightage	External exam marks
Group A: multiple choice*	20	20	20×1 = 20	20%	12
Group B: Short answer type questions	11 questions	8	8×5 = 40	40%	24
Group C: Long answer type question/case studies	6 questions	4	4×10 =40	40%	24
			100	100%	60

*Scoring scheme will not follow negative marking.