CH. DEVI LAL STATE INSTITUTE
of Engineering & Technology
PANNIWALA MOTA, SIRSA

Laboratory Manual

of

Programming for Problem Solving

Prepared By:

Anju Godara
Assistant Professor
(Computer Science & Engg. Deptt.)

PROGRAMMING FOR PROBLEM SOLVING LAB

B.Tech. 1st Year I Sem.

Note:The programs may be executed using any available Open Source/ Freely available IDE.

Course Objectives: The students will learn the following:

- To work with an IDE to create, edit, compile, run and debug programs
- To analyze the various steps in program development.
- To develop programs to solve basic problems by understanding basic concepts in Clike work with operators and control statements etc.
- To develop modular, reusable and readable C Programs using the concepts likefunctions, arrays etc.
- To Write programs using the Dynamic Memory Allocation concept.
- To create, read from and write to text and binary files.

Course Outcomes: The candidate is expected to be able to:

- Formulate the algorithms for simple problems
- Translate given algorithms to a working and correct program
- Correct syntax errors as reported by the compilers
- Identify and correct logical errors encountered during execution
- Represent and manipulate data with arrays, strings and structures
- Use pointers of different types
- Create, read and write to and from simple text and binary files
- Modularize the code with functions so that they can be reused

Hardware Requirement: Desktop Computer / laptop computer.

Software Requirement: Linux Operating System with GCC / TURBO C inWINDOWS OS / TURBO C++ in WINDOWS OS.

Some of the Tools available are:

- CodeLite: https://codelite.org/
- Code::Blocks: http://www.codeblocks.org/
- DevCpp : http://www.bloodshed.net/devcpp.html
- Eclipse: http://www.eclipse.org

INSTRUCTIONS TO STUDENTS FOR PREPARING PROGRAMMING IN C LAB REPORT

This lab manual is prepared to help the first year students with their practical understanding and development of programming skills , and may be used as a base reference during the lab practical classes.

Students have to submit Lab Exercise report of previous lab into corresponding next lab, and can be collected back after the instructor/course co-ordinator after it has been checked and signed.At the end of the semester, students should compile all the lab Excerise reports into single report and submit during end semester sessional examinations.

List of Practical/Programs to be performed in Lab

Sr. No	Module Name	Name of Program
1.	**Familiarization with programmingenvironment** Introduction to Programming, Writing of Algorithms, Introduction to Drawing flow Charts /Preparation of Flowchart/ Steps for Writing Code in C/ Turbo C.	1. First Basic Program-Writing a Single Statement. 2. Writing a Program to print your Basic details Multi statements.
2	**Variable types and type conversions**	1. WAP to perform simple Input-Output Operations in C. 2. WAP to add two numbers. 3. WAP to perform simple arithmetic operations in C(Addition, Subtraction, Multiplication, Division, Modulus). 4. Write a simple program that print the result of all the operators available in c (including pre/ post increment, bitwise and logical). 5. WAP to find area and perimeter of circle. 6. WAP to find area and perimeter of rectangle. 7. Given the values of three variable entered by user, write a program to compute anddisplay the value of x, where x=a/(b-c). 8. Write a program to convert one data type to another using auto conversion and casting. Take the value from user input.
3	**Branching and logical expressions**: Use of If, if- else, Else if, nested ifstatements and operators with them andswitch case statement	1. WAP to find whether a given number is positive or not. 2. WAP to find greatest of two numbers. 3. WAP to find greatest of three numbers using nested if/else if statements only. 4. WAP to find greatest of three numbers using & operator. 5. WAP to find whether a given number is even or odd. 6. Given the marks of a student studying five different subject. Calculate average marks of students and assign him/her Grade based on following:Marks is equal or greater than 90 – Grade A Marks equal or more than 75 and less than 90 –Grade B Marks equal or more than 60 and less than 75 –Grade C Marks equal or more than 50 and less than 60 –Grade D Marks less than 50 –Grade F 7. WAP to find roots of a quadratic equation: ax2+bx+c=0 8. WAP to print day of a week using switch case statement 9. WAP to design a simple calculate using switch-case statements
4.	**Loops: do, while and for loops:** Use of while loop, do while, and for loop:their Syntax	1. WAP to print counting 1 to 10 using all loop 2. WAP to print table of any number. 3. WAP to print the factorial of given number. 4. WAP to print the sum of digits of a given number.

		5. WAP to print the Fibonacci series up to 10 level. 6. WAP to find whether the given number is Armstrong or Not. 7. WAP to find whether the given number is Palindrome or Not. 8. WAP to find whether the given number is prime or not. 9. WAP to reverse the digits of a given number.
5.	**1D Arrays, 2 D array** Declaration of arrays, syntax, semantics,	1. Program to insert 5 elements into array and print elements of array. 2. WAP to merge two sorted array in one sorted array. 3. WAP to add two matrices in 2-D array 4. WAP to multiply two matrices in 2-D array. 5. WAP to find transpose of a Matrix. 6. WAP to find average of 10 numbers using array. 7. WAP to print the following numbers in reverse order using array.
6.	**Functions** Simple function declaration, definition,functions with return type, call by value.	1. WAP to create function display a simple message. 2. WAP to create function to add two numbers. 3. WAP to create a function to swap two numbers using call by value. 4. WAP to generate Fibonacci series using recursive function. 5. WAP to swip two integers using call by value and call by reference method of passing arguments to a function.
7.	**Pointers** Pointer declaration, use of pointers in array, functions, call by reference, recursive functions	1. WAP to understand basic use of pointers. 2. WAP to implement call by reference for swapping of two numbers. 3. WAP to calculate factorial of a number using recursion. 4. WAP to Fibonacci series up to 20 using recursive numbers
8.	**Structure** Basic of Structure, Union and accessingdata of structure.	1. WAP for user defined data type namely Student and implement it using Structure 2. WAP for user defined data type namely Book and implement it using Structure.. 3. WAP to create an array of structure.
9	**File Operations** File opening modes, creation of files,reading and writing data files.	1. WAP to read a simple file using file handling. 2. WAP to write data in file. 3. WAP to append data in existing file.
10.	**Searching and sorting** Various searching and sorting algorithms.	1. WAP to implement linear search 2. WAP to implement binary search 3. WAP to implement selection sort. 4. WAP to implement insertion sort. 5. WAP to implement quick sort. 6. WAP to implement merge sort. 7. WAP to implement bubble sort.

Suggested Reference Books for solving the problems:

i. Byron Gottfried, Schaum's Outline of Programming with C, McGraw-Hill

ii. B.A. Forouzan and R.F. Gilberg C Programming and Data Structures, Cengage Learning,(3^{rd} Edition)

iii. Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Prentice

iv. Programming in C, Stephen G. Kochan, Fourth Edition, Pearson Education.

PREFACE

This manual was developed specifically for fresh students taking up their first course in programming. Its aim is to supplement classroom lectures by focusing on C programming. Topics are arranged based on the order of class discussion. Students will be exposed to several new aspects of programming. Students will get knowledge about the concepts in C programming which includes functions, arrays, pointers, strctures e.t.c.

Practice sessions:-

1.Module Name:-Familiarization with Programming Environment

Introduction to C Programming:-

Programming is the process that professionals use to write code that instructs how a computer, application or software program performs. At its most basic, computer programming is set of instructions to Computer facilitate specific actions. Computer can do amazing things, from basic laptops capable of simple word processing and spreadsheet functions to incredibly complex supercomputers completing millions of financial transactions a day and controlling the infrastructure that makes modern life possible but no computer can do anything until a computer programmer tells it to behave in specific ways. That is what computer programming is all about .Compiler is used for performing translation.

C is a procedural programming language. It was initially developed by Dennis Ritchie in the year 1972. It was mainly developed as a system programming language to write an operating system. The main features of the C language include low-level memory access, a simple set of keywords, and a clean style these features make C language suitable for system programs like an operating system or compiler development.

C basic syntax: syntax refers to the rules that specify the correct combined sequence of symbols that can be used to form a correctly structured program using a given programming language. Programmers communicate with computers through the correctly structured syntax, semantics and grammar of a programming language.

Basic structures of c language:

```
#include<stdio.h>                    header file
#include<conio.h>
Void main()          ------------>   main method declaration
{                    ------------>   opening bracket
Printf("Add")        ------------>   body
getch()
}                    ------------>   closing brackets
```

#include <stdio.h>	This is a preprocessor statement that includes standard input output header file (stdio.h) from the C library. By including header file, you can use many different functions such as printf()
void main()	The execution of the C program starts with main() function. "void" is the return type of the

	function. Every C program can have only one main function.
Braces {}	Two curly brackets {} are used to group all code statements. This indicates begins & ends of the main function.
/* Comments */	Comments are just used to document or explain the code, to help others understand it. Everything written inside the command /* */ will be ignored by the compiler.
printf()	printf() is a function in C, which prints the text output on the screen.
getch()	This is used to read any character input from the keyboard.

Sample Program for Practice:-

1) First basic c program writing using single statement.
2) Write a program to print your basic details in multi statements.

2 Module:- Variable Type and Type Conversion

Operators in C:- An operator is a symbol that operats on a value or a variable.For example + is an operator to perform addition. C has a wide range of operators to perform various operations.

- Airthmatic Operators
- Bitwise Operators
- Relational Operators
- Logical Operators
- Assignment Operators

Basic arithmetic operation

Start.

Enter (assigne) value of the variable (addend) X.

Enter (assigne) value of the variable (addend) Y.

Calculate Z (Z = X + Y)

Display result of addition, value of Z on the monitor with the message "The sum of X and Y is:"

1. Start.

2. Initialize int a,b,c,d,e,f;

3. Accept two numbers a and b from the user.

4. Perform addition and store the result in c.

5. Perform subtraction and store the result in d.

6. Perform division and store the result in e.

7. Perform multiplication and store the result in f.

8. Stop.

Example:-variable **A** holds 10 and variable **B** holds 20 then −

Operator	**Description**	**Example**
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is true.

Relational Operators:-

Relational operators are used in decision making and loops.

Operator	Meaning of Operator	Example
==	Equal to	5 == 3 is evaluated to 0
>	Greater than	5 > 3 is evaluated to 1
<	Less than	5 < 3 is evaluated to 0
!=	Not equal to	5 != 3 is evaluated to 1
>=	Greater than or equal to	5 >= 3 is evaluated to 1
<=	Less than or equal to	5 <= 3 is evaluated to 0

Bitwise Operators:-

Operators	Meaning of operators
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise complement
<<	Shift left
>>	Shift right

Logical Opertors:-

Operators	**Example/Description**
&& (logical AND)	(x>5)&&(y<5) It returns true when both conditions are true
|| (logical OR)	(x>=10)||(y>=10) It returns true when at-least one of the condition is true

! (logical NOT)	!((x>5)&&(y<5)) It reverses the state of the operand "((x>5) && (y<5))" If "((x>5) && (y<5))" is true, logical NOT operator makes it false

Sample Program

3) Write a simple program that prints the results of all the operators available in C (including pre/post increment, bitwise and/or/not, etc.). Read required operand values from standard input.

Data Type:-

Each variable in C has an associated data type. Each data type requires different amounts of memory and has some specific operations which can be performed over it. Let us briefly describe them one by one:
Following are the examples of some very common data types used in C:

- **char:** The most basic data type in C. It stores a single character and requires a single byte of memory in almost all compilers.
- **int:** As the name suggests, an int variable is used to store an integer.
- **float:** It is used to store decimal numbers (numbers with floating point value) with single precision.
- **double:** It is used to store decimal numbers (numbers with floating point value) with double precision.

Different data types also have different ranges upto which they can store numbers. These ranges may vary from compiler to compiler.

Data Type	Memory (bytes)	Range	Format Specifier
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld

Data Type	Memory (bytes)	Range	Format Specifier
unsigned long int	4	0 to 4,294,967,295	%lu
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4	--	%f
double	8	--	%lf
long double	16	--	%Lf

Sample Program:-

4) Write a simple program that converts one given data type to another using auto conversion andcasting. Take the values form standard input.
5) WAP to find area and perimeter of circle.
6) WAP to find area and perimeter of rectangle.
7) Given the values of three variable entered by user, write a program to compute anddisplay the value of x, where x=a/(b-c).
8) Write a program for find the max and min from the three numbers.

3 Module Branching & Logical expressions:-

If you put some condition for a block of statements, the execution flow may change based on

the result evaluated by the condition. This process is called decision making in 'C. ' It is also called as branching as a program decides which statement to execute based on the result of the evaluated condition.

If Statement:-

The if statement evaluates the test expression inside the parenthesis () . If the test expression is evaluated to true, statements inside the body of if are executed. If the test expression is evaluated to false, statements inside the body of if are not executed.

Syntax:-

```
If (condition)
{
     //Block of C statements here
    //These statements will be execute if the condition is true
}
```

If else Statements:-

The if-else is statement is an extended version of If. The general form of if-else is as follows:

Syntax:-

```
if (test-expression)
{
   True block of statements
}
Else
{
   False block of statements
}
Statements;
```

Nested Else-if statements:-

Nested else-if is used when multipath decisions are required.

The general syntax of how else-if ladders are constructed in ‘C’ programming is as follows:

Syntax:-

```
 if (test - expression 1) {
    statement1;
} else if (test - expression 2) {
    Statement2;
} else if (test - expression 3) {
    Statement3;
} else if (test - expression n) {
    Statement n;
} else {
```

default;
}
Statement x;
This type of structure is known as the else if ladder.

Switch Statements:-

The switch statement allows us to execute one code block among many alternatives.

You can do the same thing with the if...else..if ladder. However, the syntax of the switch statement is much easier to read and write.

Syntax:-

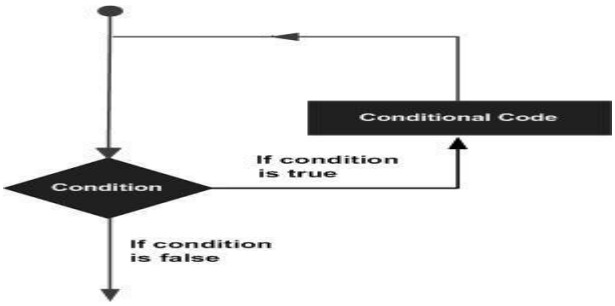
```
Switch (expression )
{

case constant1:

//statements
break;

case constant2:
// statements
break;
.

default:
//default statements
  }
```

Sample Programs:-

1) Program to find whether a given number is positive or not.
2) Program to find greatest of two numbers.
3) Program to find greatest of three numbers using nested if/else if statements only.
4) Program to find greatest of three numbers using & operator.
5) Program to find whether a given is divisible by 5 and 11.
6) Program to find whether a given number is even or odd.
7) Program to check the validity of a triangle when three angles are given.
8) Given the marks of a student studying five different subject. Calculate average marks ofstudents and assign him/her Grade based on following:Marks is equal or greater than 90 – Grade AMarks equal or more than 75 and less than 90 –Grade BMarks equal or more than 60 and less than 75 –Grade CMarks equal or more than 50 and less than 60 –Grade D Marks less than 50 –Grade F
9) Program to print day of a week using switch case statement.
10) Program to design a simple calculate using switch-case statements.

4 Module Loops:-

A loop statement allows us to execute a statement or group of statements multiple times. Given below is the general form of a loop statement in most of the programming languages –



C programming language provides the following types of loops to handle looping requirements.

1 While loop-

Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.

Syntax:

```
While(test condition)
{
    Body of the loop
}
```

2 For loop-

Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

Syntax:-

```
For(initialization; test condition; increment or decrement)
{
    Body of the loop
}
```

3 do...while loop:-

It is more like a while statement, except that it tests the condition at the end of the loop body.

Syntax:-

```
do
{
    body of the loop
}
While (test condition);
```

Sample Program:-

1) Program to print counting 1 to 10 using all loop
2) Program to print table of any number.
3) Program to print the factorial of given number.
4) Program to print the sum of digits of a given number.
5) Program to print the Fibonacci series up to 10 level.
6) Program to find whether the given number is Armstrong or Not.
7) Program to find whether the given number is Palindrome or Not.
8) Program to find whether the given number is prime or not.
9) Program to reverse the digits of a given number.

5 Module Array:-

The syntax is the same as for a normal variable declaration except the variable name should be followed by subscripts to specify the size of each dimension of the array. Array is defined as **the collection of similar type of data items stored at contiguous memory locations**. Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc. The general form for an array declaration would be:-

1) 1-D Array
2) 2-D array

1-D Array-
One dimensional array is an array that has only one subscript specification that is needed to specify a particular element of an array. A one-dimensional array is a structured collection of components (often called array elements) that can be accessed individually by specifying the position of a component with a single index value.
Syntax:-data-type arr_name[array_size];

2- DArray-
The two-dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns. However, 2D arrays are created to implement a relational database lookalike data structure. It provides ease of holding the bulk of data at once which can be passed to any number of functions wherever required.
Syntax:-data_type array_name[rows][columns];

Sample Program:-

1) Program to insert 5 elements into array and print elements of array.
2) Program to perform addition of two matrix.
3) Program to perform multiplication two matrix
4) Program to perform Transpose of 2 matrix.
5) Program to find average of 10 number using array.
6) Program to print the following numbers in reverse order using array.

6 Module Functions:-

In c, we can divide a large program into the basic building blocks known as function. The function contains the set of programming statements enclosed by {}. A function can be called multiple times to provide reusability and modularity to the C program. In other words, we can say that the collection of functions creates a program. The function is also known as procedure *or* subroutine *in* other programming languages.
Syntax:-

SN	C function aspects	Syntax
1	Function declaration	return_type function_name (argument list);
2	Function call	function_name (argument_list)
3	Function definition	return_type function_name (argument list) {function body;}

Syntax:-

```
return_type function_name(data_type parameter...)
{
//code to be executed
}
```

Types of Functions:-

There are two types of functions in C programming:

1. **Library Functions**: are the functions which are declared in the C header files such as scanf(), printf(), gets(), puts(), ceil(), floor() etc.
2. **User-defined functions**: are the functions which are created by the C programmer, so that he/she can use it many times. It reduces the complexity of a big program and optimizes the code.

Sample Program:-

1) Program to create function to add two numbers.
2) Program to create a function to swap two numbers using call by value.

3) Program to generate Fibonacci series using recursive function.
4) Program to swap two integers using call by value and call by referencemethods of passing arguments to a function.

7 Module Pointers:-

The pointer in C language is a variable which stores the address of another variable. This variable can be of type int, char, array, function, or any other pointer. The size of the pointer depends on the architecture. By the help of * (**indirection operator**), we can print the value of pointer variable p.

Syntax:-

```
data_ type * pointer_ name;
```

Sample Program:-

1) Program to understand basic use of pointers.
2) Program to implement call by reference for swapping of two numbers.
3) Program to calculate factorial of a number using recursion.
4) Program to Fibonacci series up to 20 using recursive numbers.

8 Module Structure:-

Structure in c is a user-defined data type that enables us to store the collection of different data types. Each element of a structure is called a member. Structures ca; simulate the use of classes and templates as it can store various information. The ,struct keyword is used to define the structure.

Syntax:-

```
 struct              tag _name
{
        data_type       member1;
        data_type       member2;

         ------------       ------------
         ------------       ------------
};
```

Sample Program:-

1) Program to find user define data type namely student and implement it using array in structure.

9 Module File Operations:-

File handling in C enables us to create, update, read, and delete the files stored on the local file system through our C program. The following operations can be performed on a file.

- Creation of the new file
- Opening an existing file
- Reading from the file

- Writing to the file
- Deleting the file

Functions for file handling:-

There are many functions in the C library to open, read, write, search and close the file. A list of file functions are given below:

No.	Function	Description
1	fopen()	opens new or existing file
2	fprintf()	write data into the file
3	fscanf()	reads data from the file
4	fputc()	writes a character into the file
5	fgetc()	reads a character from file
6	fclose()	closes the file
7	fseek()	sets the file pointer to given position
8	fputw()	writes an integer to file
9	fgetw()	reads an integer from file
10	ftell()	returns current position
11	rewind()	sets the file pointer to the beginning of the file

Sample Program:-

1) WAP to read a simple file using file handling.
2) WAP to write data in file.
3) WAP to append data in existing file.

10 Module Searching and Sorting:-

Searching Algorithms are designed to check for an element or retrieve an element from any data structure where it is stored.

Linear Search:-

A linear search, also known as a sequential search, is **a method of finding an element within a list**. It checks each element of the list sequentially until a match is found or the

whole list has been searched.

Algorithm:-
Linear Search (Array A, Value x)

Step 1: Set i to 1
Step 2: if i > n then go to step 7
Step 3: if A[i] = x then go to step 6
Step 4: Set i to i + 1
Step 5: Go to Step 2
Step 6: Print Element x Found at index i and go to step 8
Step 7: Print element not found
Step 8: Exit

Binary search:-

Binary search is a fast search algorithm with run-time complexity of O(log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form.

Binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of item is returned. If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item. Otherwise, the item is searched for in the sub-array to the right of the middle item. This process continues on the sub-array as well until the size of the subarray reduces to zero.

Procedure binary_search:-

```
  A ← sorted array
  n ← size of array
  x ← value to be searched

  Set lowerBound = 1
  Set upperBound = n

  while x not found
    if upperBound < lowerBound
      EXIT: x does not exists.

    set midPoint = lowerBound + ( upperBound - lowerBound ) / 2

    if A[midPoint] < x
      set lowerBound = midPoint + 1

    if A[midPoint] > x
      set upperBound = midPoint - 1

    if A[midPoint] = x
      EXIT: x found at location midPoint
  end while

end procedure
```

Sample Program:-

1) Write a program that uses function to search for a key value in a given list of integer using linear search method.
2) Write a program that uses function to search for a key value in a given sorted list of integer using Binary search method.

Sorting:-

Sorting refers to arranging data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order. Most common orders are in numerical or lexicographical order.

Bubble Sort:-

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$ where **n** is the number of items.

```
begin BubbleSort(list)

   for all elements of list
      if list[i] > list[i+1]
         swap(list[i], list[i+1])
      end if
   end for

   return list

end BubbleSort
```

Insertion sort:-

This is an in-place comparison-based sorting algorithm. Here, a sub-list is maintained which is always sorted. For example, the lower part of an array is maintained to be sorted. An element which is to be 'insert'ed in this sorted sub-list, has to find its appropriate place and then it has to be inserted there. Hence the name, **insertion sort**.

The array is searched sequentially and unsorted items are moved and inserted into the sorted sub-list (in the same array). This algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$, where **n** is the number of items.

Step 1 − If it is the first element, it is already sorted. return 1;
Step 2 − Pick next element
Step 3 − Compare with all elements in the sorted sub-list
Step 4 − Shift all the elements in the sorted sub-list that is greater than the value to be sorted
Step 5 − Insert the value

Step 6 − Repeat until list is sorted

Selection Sort:-

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

This algorithm is not suitable for large data sets as its average and worst case complexities are of $O(n^2)$, where **n** is the number of items.

Algorithm:-

Step 1 − Set MIN to location 0
Step 2 − Search the minimum element in the list
Step 3 − Swap with value at location MIN
Step 4 − Increment MIN to point to next element
Step 5 − Repeat until list is sorted

Merge Sort:-

Merge sort is a sorting technique based on divide and conquer technique. With worst-case time complexity being O(n log n), it is one of the most respected algorithms.Merge sort first divides the array into equal halves and then combines them in a sorted manner.

Step 1 − if it is only one element in the list it is already sorted, return.
Step 2 − divide the list recursively into two halves until it can no more be divided.
Step 3 − merge the smaller lists into new list in sorted order.

Quick sort:-

Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays. A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value. Quick sort partitions an array and then calls itself recursively twice to sort the two resulting subarrays. This algorithm is quite efficient for large-sized data sets as its average and worst-case complexity are $O(n^2)$, respectively.

Step 1 − Choose the highest index value has pivot
Step 2 − Take two variables to point left and right of the list excluding pivot
Step 3 − left points to the low index
Step 4 − right points to the high
Step 5 − while value at left is less than pivot move right
Step 6 − while value at right is greater than pivot move left
Step 7 − if both step 5 and step 6 does not match swap left and right

Step 8 − if left ≥ right, the point where they met is new pivot

Sample Programs:-

1) Write a C program that implements the Bubble sort method to sort a given list of integers in ascending/ descending order.
2) Write a C program that sorts the given array of integers using selection sort in descendingorder
3) Write a C program that sorts the given array of integers using insertion sort in ascendingorder
4) Write a c program that sorts the given array of integers using quick sort and merge sort.