



CMR College of Engineering & Technology
Kandlakoya(v), Medchal Road Hyderabad, Telangana,
India -501401,
Telephone: 08418 - 200699. Email: info@cmrcet.ac.in.



C PROGRAMMING & DATA STRUCTURES LABORATORY

B. TECH: I YEAR – I SEMESTER (2022-2023)

CMR College of Engineering & Technology

Vision

Our Vision is to remain a premier academic institution striving continuously for excellence in technical education, research and render technological services to the nation.

Mission

- Our Mission is to create and sustain a community of learning in which students acquire knowledge and learn to apply it professionally with a concern for the society.
- Pursue and Disseminate Research Findings and Offer Knowledge-Based Technological Services to Satisfy the Needs of Society and the Industry.
- Promote Professional Ethics, Leadership Qualities and Social Responsibilities.

Vision of the Department

- To evolve as a centre of academic excellence in Computer Science & Engineering by building strong teaching and research environment.

Mission of the Department

- To offer high quality graduate and post graduate programs in computerscienceeducation and to prepare students for professional career and/orhigher studies globally.
- To develop self-learning abilities and professional ethics to serve thesociety.

Program Educational Objectives (PEOs)

| | |
|-----------|--|
| PEO - I | Excel in their professional career and higher education in Computer Science & Engineering and chosen fields. |
| PEO - II | Demonstrate leadership qualities, team work and professional ethics to serve the society |
| PEO - III | Adapt to state of art technology through continuous learning in the areas of interest. |



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CMR College of Engineering & Technology
Kandlakoya(v), Medchal Road Hyderabad, Telangana, India -
501401, Telephone: 08418 - 200699. Email: info@cmrcet.ac.in.

Academic Year - 2022-23

Semester –I

Subject : C PROGRAMMING & DATA STRUCTURES LABORATORY

Subject Code : A405503

Class & Branch/ Specialization: I B.Tech. I – Semester
(Common to ECE, EEE, Mech and Civil)

LAB OBJECTIVES

| S.No. | Lab Objectives |
|--------------|--|
| 1 | Work with an IDE to create, edit, compile, run and debug programs |
| 2 | Analyze the various steps in program development |
| 3 | develop programs to solve basic problems by understanding basic concepts in C like operators, control statements etc |
| 4 | Develop modular, reusable and readable C Programs using the concepts like functions, arrays etc. |
| 5 | Write programs using the Dynamic Memory Allocation concept |
| 6 | create, read from and write to text and binary files |

Signature of the HOD



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR College of Engineering & Technology

Kandlakoya(v), Medchal Road Hyderabad, Telangana,

India - 501401, Telephone: 08418 - 200699.

Email: info@cmrcet.ac.in.

Academic Year - 2022-23

Semester –I

Subject : C PROGRAMMING & DATA STRUCTURES LABORATORY

Subject Code : A405503

Class & Branch/ Specialization: I B.Tech I-Sem

(Common to CSE, IT, CSE-DS, CSE-AI&ML, AI&ML, AI&DS, CSE-CS)

COURSE (LAB) OUTCOMES

| S.No. | Course (Lab) Outcomes |
|-------|---|
| 1 | Formulate the algorithms for simple problems and translate given algorithms to a working and correct program |
| 2 | Correct syntax errors as reported by the compilers identify and correct logical errors encountered during execution |
| 3 | Develop applications using pointer concept. |
| 4 | create, read and write to and from simple text and binary files |
| 5 | Develop reusable code with the help C-functions |

Signature of the HOD

SYLLABUS

Objectives:

1. To understand the various steps in program development.
2. To understand the basic concepts in C Programming Language.
3. To understand different modules that includes conditional and looping expressions.
4. To understand how to write modular and readable C Programs.
5. To write programs in C to solve problems using arrays, structures and files.

| WEEK | NAME OF THE PROGRAM |
|----------|--|
| WEEK - 1 | <p style="text-align: center;">I. OPERATORS AND EVALUATION OF EXPRESSIONS</p> <p>Demonstration</p> <p>1. Write a C program to print greetings message on the screen. 2. Write a C program to illustrate usage of comments in C.</p> <p>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.</p> <p>4. Write a C program that converts given data type to another using auto conversion and casting. Take the values from standard input.</p> <p>5. Write a program for finding the max and min from the three numbers (using ternary operator).</p> |
| WEEK - 2 | <p>Experiment</p> <p>6. Write a C program to compute simple, compound interest.</p> <p>7. Write a C program that declares Class awarded for a given percentage of marks, where mark = 70% = Distinction. (Read percentage from standard input).</p> |
| WEEK - 3 | <p>II. Expression Evaluation</p> <p>Demonstration</p> <p>1. A building has 10 floors with a floor height of 3 meters each. A ball is dropped from the top of the building. Find the time taken by the ball to reach each floor. (Use the formula $s = ut + \frac{1}{2}at^2$ where u and a are the initial velocity in m/sec ($= 0$) and acceleration in m/sec^2 ($= 9.8 m/s^2$)).</p> |

| | |
|----------|---|
| | <p>2. Write a program that asks the user to enter the highest rainfall ever in one season for a country, and the rainfall in the current year for that country, obtains the values from the user, checks if the current rainfall exceed the highest rainfall and prints an appropriate message on the screen. If the current rainfall is higher, it assigns that value as the highest rainfall ever. Use only the single-selection form of the if statement.</p> |
| WEEK - 4 | <p>Experiment</p> <p>3. Write a C program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.</p> <p>4. Write a C program to find the roots of a Quadratic equation</p> |
| WEEK - 5 | <p>III. Iterative statements</p> <p>Demonstration</p> <p>1. Write a program that reads an integer (5 digits or fewer) and determines and prints how many digits in the integer are 9s.</p> <p>2. Write a program that keeps printing the powers of the integer 3, namely 3, 9, 27, 91, 273, and so on. Your loop should not terminate (i.e., you should create an infinite loop). What happens when you run this program?</p> <p>3. Write a program that reads the radius of a circle (as a float value) and computes and prints the diameter, the circumference and the area. Use the value 3.14159 for π</p> |
| WEEK - 6 | <p>Experiment</p> <p>4. Write a menu driven C program that allows a user to enter n numbers and then choose between finding the smallest, largest, sum, or average. The menu and all the choices are to be functions. Use a switch statement to determine what action to take. Display an error message if an invalid choice is entered.</p> <p>5. Write a C program to construct a pyramid of numbers as follows:</p> <pre> 1 1 1 2 2 2 1 2 3 3 3 3 1 2 3 4 4 4 4 4 1 2 3 4 5 5 5 5 5 5 </pre> |

| | |
|-----------|--|
| | |
| WEEK - 7 | <p>IV. Arrays, Pointers, and Functions</p> <p>Demonstration</p> <ol style="list-style-type: none"> 1. Write a C program to find the minimum, maximum and average in an array of integers. 2. Write a function to compute mean, variance, Standard Deviation, sorting of n elements in a single dimension array. 3. Write a C program that uses functions to perform the following: i. Addition of Two Matrices ii. Multiplication of Two Matrices iii. Transpose of a matrix. |
| WEEK - 8 | <p>Experiment</p> <ol style="list-style-type: none"> 4. Write a C program to find the GCD (greatest common divisor) of two given integers. 5. Write a C program to compute x^n |
| WEEK - 9 | <p>V. Strings</p> <p>Demonstration</p> <ol style="list-style-type: none"> 1. Write a C program to convert a Roman numeral ranging from I to L to its decimal equivalent. 2. Write a C program that converts a number ranging from 1 to 50 to Roman equivalent c. 3. Write a C program that uses functions to perform the following operations: <ul style="list-style-type: none"> • To insert a sub-string into a given main string from a given position. • To delete n Characters from a given position in a given string. |
| WEEK - 10 | <p>Experiment</p> <ol style="list-style-type: none"> 4. Write a C program to determine if the given string is a palindrome or not (Spelled same in both directions with or without a meaning like madam, civic, noon, abcba, etc.) 5. Write a C program that displays the position of a character ch in the string S or – 1 if S doesn't contain ch. 6. Write a C program to count the lines, words and characters in a given text. |
| WEEK - 11 | VI Data Structures |

| | |
|-----------|--|
| | <p>Demonstration</p> <ol style="list-style-type: none"> 1. Write a program that uses functions to perform the following operations on singly linked list i) Creation ii) Insertion iii) Deletion iv) Traversal 2. Write a program that implement stack (its operations) using i) Arrays ii) Pointers 3. Write a program that implement Queue (its operations) using i) Arrays ii) Pointers |
| WEEK - 12 | <p>Experiment</p> <ol style="list-style-type: none"> 4. Write a program that uses functions to perform the following operations on doubly linked List. i) Creation ii) Insertion iii) Deletion iv) Traversal 5. Write a program that uses functions to perform the following operations on circular linked List. i) Creation ii) Insertion iii) Deletion iv) Traversal |
| WEEK - 13 | <p>VII Searching & Sorting Demonstration</p> <ol style="list-style-type: none"> 1. Write a C program that uses non recursive function to search for a Key value in a given list of integers using linear search method. 2. Write a C program that uses non recursive function to search for a Key value in a given sorted list of integers using binary search method. 3. Write a C program that implements the Bubble sort method to sort a given list of integers in ascending order. |
| WEEK - 14 | <p>Experiment</p> <ol style="list-style-type: none"> 4. Write a C program that sorts the given array of integers using selection sort in descending order 5. Write a C program that sorts the given array of integers using insertion sort in ascending order |
| PROJECTS | <ol style="list-style-type: none"> 1. Library management system 2. Payrol management system 3. Telecom billing management system 4. Bank management system 5. Employee's management system 6. Library management system 7. Personal Diary management system |

| | |
|--|--|
| | 8. Medical store management system. 9. Phone Contacts management 10. Fee Collection system |
|--|--|

TEXTBOOKS:

1. Jeri R. Hanly and Elliot B. Koffman, Problem solving and Program Design in C7th Edition, Pearson
2. B.A. Forouzan and R.F. Gilberg C Programming and Data Structures, Cengage Learning, (3rd Edition)

REFERENCE BOOKS:

1. Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, PHI
2. E. Balagurusamy, Computer fundamentals and C, 2nd Edition, McGraw-Hill
3. Yashavant Kanetkar, Let Us C, 18th Edition, BPB
4. R.G. Dromey, How to solve it by Computer, Pearson (16th Impression)
5. Programming in C, Stephen G. Kochan, Fourth Edition, Pearson Education.
6. Herbert Schildt, C: The Complete Reference, McGraw Hill, 4th Edition
7. Byron Gottfried, Schaum's Outline of Programming with C, McGraw-Hill

INDEX

| S.NO | NAME OF THE PROGRAM | P.NO |
|------|--|------|
| 1 | Write a C program to print greetings message on the screen. | |
| 2 | Write a C program to illustrate usage of comments in C. | |
| 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. | |
| 4 | Write a C program that converts given data type to another using auto conversion and casting. Take the values from standard input. | |
| 5 | Write a program for finding the max and min from the three numbers (using ternary operator). | |
| 6 | Write a C program to compute simple, compound interest. | |
| 7 | Write a C program that declares Class awarded for a given percentage of marks, where mark = 70% = Distinction. (Read percentage from standard input). | |
| 8 | A building has 10 floors with a floor height of 3 meters each. A ball is dropped from the top of the building. Find the time taken by the ball to reach each floor. (Use the formula $s = ut + \frac{1}{2}at^2$ where u and a are the initial velocity in m/sec ($= 0$) and acceleration in m/sec^2 ($= 9.8 m/s^2$)). | |
| 9 | Write a program that asks the user to enter the highest rainfall ever in one season for a country, and the rainfall in the current year for that country, obtains the values from the user, checks if the current rainfall exceed the highest rainfall and prints an appropriate message on the screen. If the current rainfall is higher, it assigns that value as the highest rainfall ever. Use only the single-selection form of the if statement. | |
| 10 | Write a C program to generate all the prime numbers between 1 and n , where n is a value supplied by the user. | |
| 11 | Write a C program to find the roots of a Quadratic equation | |
| 12 | Write a program that reads an integer (5 digits or fewer) and determines and prints how many digits in the integer are 9s. | |

| | | |
|----|--|--|
| | | |
| 13 | Write a program that keeps printing the powers of the integer 3, namely 3, 9, 27, 91, 273, and so on. Your loop should not terminate (i.e., you should create an infinite loop). What happens when you run this program? | |
| 14 | Write a program that reads the radius of a circle (as a float value) and computes and prints the diameter, the circumference and the area. Use the value 3.14159 for π | |
| 15 | Write a menu driven C program that allows a user to enter n numbers and then choose between finding the smallest, largest, sum, or average. The menu and all the choices are to be functions. Use a switch statement to determine what action to take. Display an error message if an invalid choice is entered. | |
| 16 | Write a C program to construct a pyramid of numbers as follows: <pre> 1 1 1 2 2 2 1 2 3 3 3 3 1 2 3 4 4 4 4 4 1 2 3 4 5 5 5 5 5 5 </pre> | |
| 17 | Write a C program to find the minimum, maximum and average in an array of integers. | |
| 18 | Write a function to compute mean, variance, Standard Deviation, sorting of n elements in a single dimension array. | |
| 19 | Write a C program that uses functions to perform the following: i. Addition of Two Matrices ii. Multiplication of Two Matrices iii. Transpose of a matrix | |
| 20 | Write a C program to find the GCD (greatest common divisor) of two given integers. | |
| 21 | Write a C program to compute x^n | |
| 22 | Write a C program to convert a Roman numeral ranging from I to L to its decimal equivalent. | |
| 23 | Write a C program that converts a number ranging from 1 to 50 to Roman equivalent c. | |

| | | |
|----|--|--|
| 24 | Write a C program that uses functions to perform the following operations: <ul style="list-style-type: none"> • To insert a sub-string into a given main string from a given position. • To delete n Characters from a given position in a given string | |
| 25 | Write a C program to determine if the given string is a palindrome or not (Spelled same in both directions with or without a meaning like madam, civic, noon, abcba, etc.) | |
| 26 | Write a C program that displays the position of a character ch in the string S or – 1 if S doesn't contain ch. | |
| 27 | Write a C program to count the lines, words and characters in a given text. | |
| 28 | Write a program that uses functions to perform the following operations on singly linked list i) Creation ii) Insertion iii) Deletion iv) Traversal | |
| 29 | Write a program that implement stack (its operations) using i) Arrays ii) Pointers | |
| 30 | Write a program that implement Queue (its operations) using i) Arrays ii) Pointers | |
| 31 | Write a program that uses functions to perform the following operations on doubly linked List. i) Creation ii) Insertion iii) Deletion iv) Traversal | |
| 32 | Write a program that uses functions to perform the following operations on circular linked List. i) Creation ii) Insertion iii) Deletion iv) Traversal | |
| 33 | Write a C program that uses non recursive function to search for a Key value in a given list of integers using linear search method. | |
| 34 | Write a C program that uses non recursive function to search for a Key value in a given sorted list of integers using binary search method. | |
| 35 | Write a C program that implements the Bubble sort method to sort a given list of integers in ascending order. | |
| 36 | Write a C program that sorts the given array of integers using selection sort in descending order | |
| 37 | Write a C program that sorts the given array of integers using insertion sort in ascending order | |

WEEK1
DEMONSTRATION
OPERATORS AND EVALUATION OF EXPRESSIONS

1. C program to print greetings message on the screen.

- if time is greater than 0 and less than or equal to 3, then print Good Night
- If time is greater than 3 and less than 12, then print Good Morning
- If time is equal to 12, then print Good Noon
- If time is greater than 12 and less than or equal to 15, print Good AfterNoon
- If time is greater than 15 and less than 20, print Good Evening
- If time is greater than or equal to 20 and less than or equal to 24, print Good Night

format:

- If time equals 1 means it is 1 A.M.
- If time equals 13 means it is 1 P.M.
- If time equals 15 means it is 3 P.M.
- If time equals 20 means it is 8 P.M.

CODE:

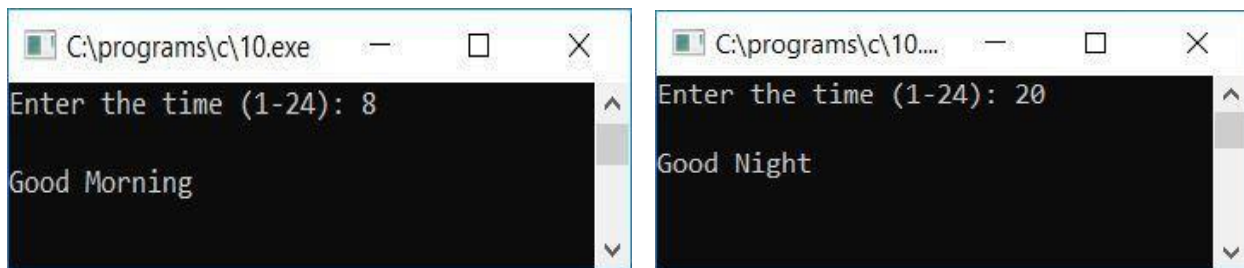
```
#include<stdio.h>
#include<conio.h>
int main()
{
int t;
printf("Enter the time (1-24): ");
scanf("%d", &t);
printf("\n");
if(t>0 && t<=3)
printf("Good Night");
else if(t>3 && t<12)
printf("Good Morning");
else if(t==12)
printf("Good Noon");
```

```

else if(t>12 && t<=15)
printf("Good AfterNoon");
else if(t>15 && t<20)
printf("Good Evening");
else if(t>=20 && t<=24)
printf("Good Night");
else
printf("Unkownn time!");
getch();
return 0;
}

```

output



2. C program to illustrate usage of comments in C.

There are two types of comments in C

Single-line Comments in C

In C, a single line comment starts with //. It starts and ends in the same line. `printf("Hello World!"); // This is a comment`

Multi-line Comments in C

`/* The code below will print the words Hello World!to the screen, and it is amazing */`
`printf("Hello World!");`

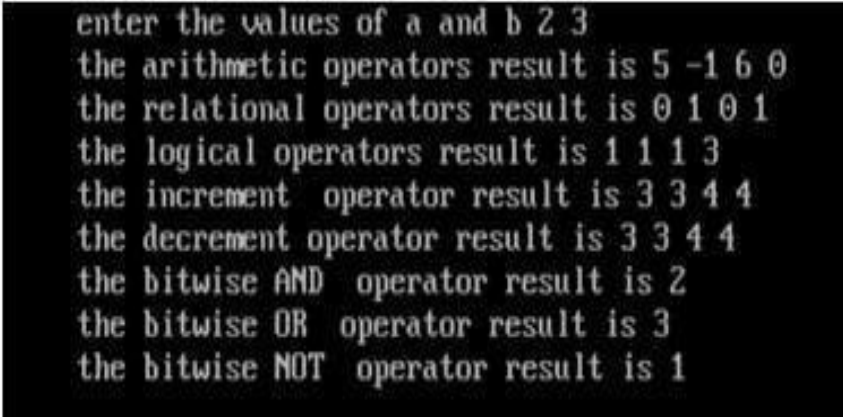
3. program that prints the results of all the operators available in C.

CODE:

```
#include<stdio.h>

void main( )
{
inta,b;
printf("enter the values of a and b");
scanf("%d%d",&a,&b);
printf("the arithmetic operators result is %d %d %d %d", a+b,a-b,a*b,a/b);
printf("the relational operators result is %d %d %d %d", a>b,a<b,a>=b,a<=b);
printf("the logical operators result is %d %d %d %d", a&& b,a||b,!(a==b));
printf("the increment operator result is %d %d %d %d",a++,++a,b++,++b);
printf("the decrement operator result is %d %d %d %d",a--,--a,b--,--b);
printf("the bitwise AND operator result is %d",a&b);
printf("the bitwise OR operator result is %d",a|b);
printf("the bitwise NOT operator result is %d",a^b);
}
```

output

A screenshot of a terminal window showing the output of a C program. The text is as follows:

```
enter the values of a and b 2 3
the arithmetic operators result is 5 -1 6 0
the relational operators result is 0 1 0 1
the logical operators result is 1 1 1 3
the increment operator result is 3 3 4 4
the decrement operator result is 3 3 4 4
the bitwise AND operator result is 2
the bitwise OR operator result is 3
the bitwise NOT operator result is 1
```


3. C program that converts given data type to another using auto conversion and casting. Take the values from standard input.

There are 2 types of type casting operations in C :-

1. Implicit Type Casting :

With the help of implicit type casting, you can convert a data type of a variable into another data type without losing its actual meaning. The conversion without changing the importance of the values stored inside the variable is called “implicit casting”

Example of Implicit Type Casting

```
int a =4;
float x = 12.4, y;
y = x / a;
```

the variable ‘a’ will be automatically converted to float data type which is a bigger data type. and the variable ‘y’ will be equal to ‘x’ in terms of its data type. So, the value of y will be $12.4/4.0=3.1$.

2. Explicit Type Casting

Explicit casting must be done manually by placing the type in parentheses in front of the value:

Example:

```
float f = 6.78;
int k = (int) f; // Manual casting: float to int
```

Program:

```
#include<stdio.h>
main()
{
float sum,count;int
mean;
printf("enter the value of sum and count");
scanf("%f%f",&sum,&count);
mean=sum/count;
printf("the value of mean is :%d\n",mean);
}
```

Output

```
enter the value of sum and count 25.45 5.00
the value of mean is : 5
```

4. Write a program for finding the max and min from the three numbers (using ternary operator).

Explanation:

Syntax of ternary operator is

(expression-1) ? expression-2 : expression-3

This operator returns one of two values depending on the result of an expression. If "expression-1" is evaluated to Boolean true, then expression-2 is evaluated and its value is returned as a final result otherwise expression-3 is evaluated and its value is returned as a final result.

Program:

```
#include <stdio.h>
int main() {
    int a, b, c, max,min ;
    /*
    * Take three numbers as input from user
    */
    printf("Enter Three Integers\n");
    scanf("%d %d %d", &a, &b, &c);

    max = (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c);
```

```
/* Print Maximum Number */ printf("Maximum  
Number is = %d\n", max);  
  
min = (a < b) ? ((a < c) ? a : c) : ((b < c) ? b : c);  
  
/* Print Maximum Number */ printf("Minimum  
Number is = %d\n", min);  
return 0;  
}
```

Output:

Enter Three Integers 12

2 13

Minimum Number is = 2

Enter Three Integers 12

2 13

Maximum Number is = 13

VIVA-QUESTIONS:

1. The format identifier '%i' is also used for _____ data type?

- A. char
- B. int
- C. float
- D. double

2. Which data type is most suitable for storing a number 65000 in a 32-bit system?

- A. short
- B. int
- C. long
- D. double

3. Which of the following is a User-defined data type?

- A. typedef int Boolean;
- B. typedef enum {Mon, Tue, Wed, Thu, Fri} Workdays;
- C. struct {char name[10], int age};
- D. All of the mentioned

4. What is the size of an int data type?

- A. 4 Bytes
- B. 8 Bytes
- C. Depends on the system/compiler
- D. Cannot be determined.

5. All keywords in C are in?

- A. Lower Case letters
- B. Upper Case letters
- C. Camel Case letters
- D. None

6. Which of the following operator takes only integer operands?

- A. +
- B. *
- C. /
- D. %

7. What is the output of the following code?

- ```
#include<stdio.h> int
main()
{
int x, y, z;
x = 9 > 8 > 7;
y = 9 > 8 > 0;
z = 9 > 8 > 1;
printf("%d %d %d", x, y, z);
return 0;
}
```
- A. 0 1 1
- B. 1 0 0
- C. 0 0 1
- D. 0 1 0

8. Which of the following is not a valid C variable name?

- A. int number;
- B. float rate;
- C. int variable\_count;
- D. int \$main;

**WEEK – 2**  
**Experiment**

**Program 6:**

**6. Write a C program to compute simple and compound interest**

**Problem Description:** C program to compute simple, compound interest

**Algorithm:**

**Step 1:**Start.

**Step 2:**Read Principal Amount, Rate and Time.

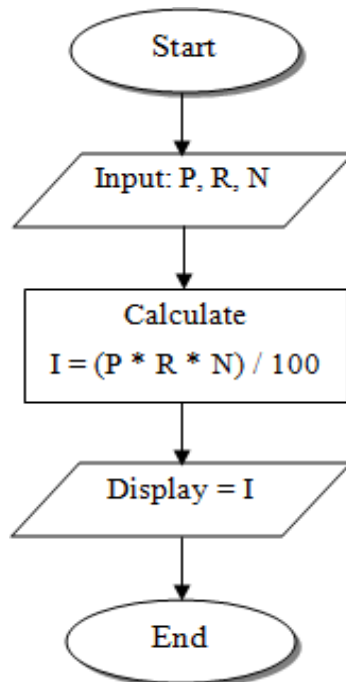
**Step 3:**Calculate Interest using formula  $SI = ((\text{amount} * \text{rate} * \text{time}) / 100)$ ,  $CI = P * (1 + r/100)^n - P$

**Step 4:**Print Simple Interest and compound interest

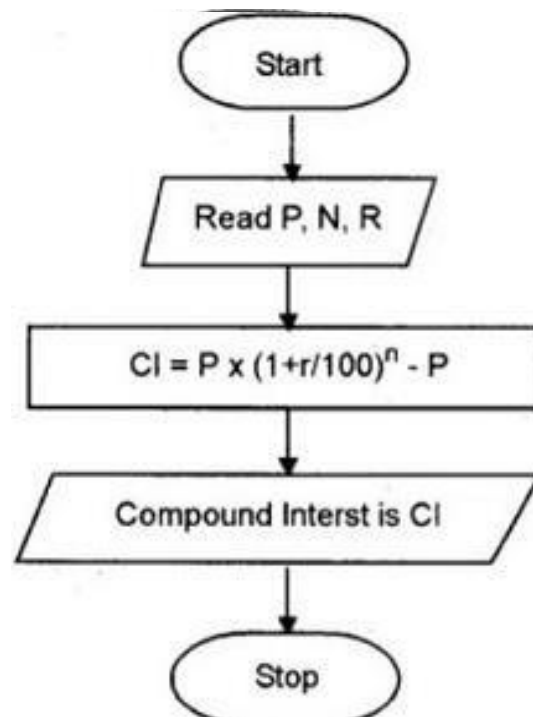
**Step 5:**Stop

**Flowchart: simple**

**interest:**



**Compound interest:**



## Program 7

**7. Write a C Program that declares Class awarded for a given percentage of marks, where mark = 70% = Distinction. (Read percentage from standard input.)**

**Problem Description:** This program declares Class awarded for a given percentage of marks, where mark = 70% = Distinction.

**Algorithm:**

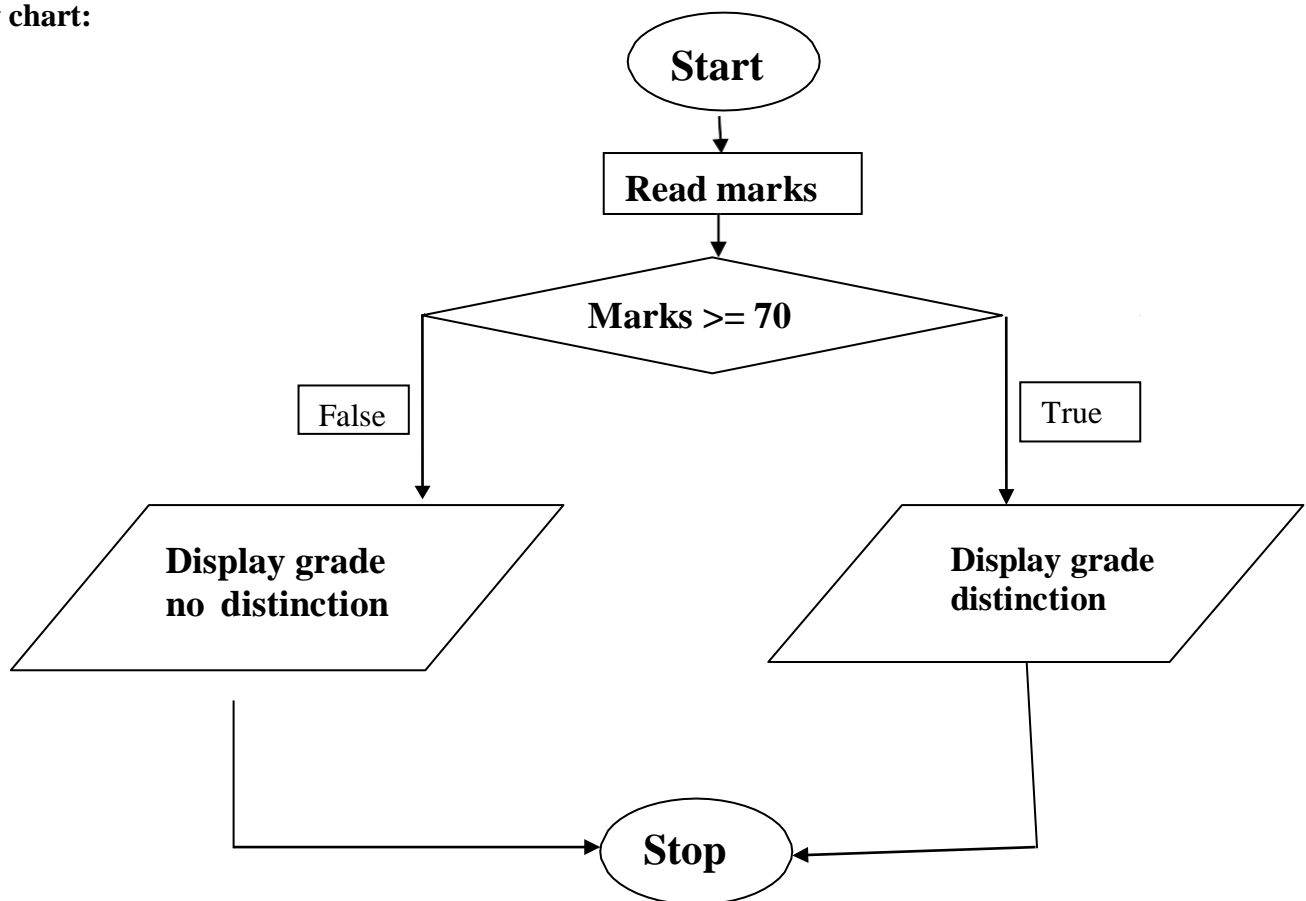
**Step 1 :** start

**Step 2 :** read marks or Percentage

**Step 3 :** if marks  $\geq$  70 then grade =distinction

**Step 4 :** stop.

**Flow chart:**



**Week-3:      Expression      Evaluation**  
**Demonstration**

**1.A building has 10 floors with a floor height of 3 meters each. A ball is dropped from the top of the building. Find the time taken by the ball to reach each floor. (Use the formula  $s = ut + (1/2)at^2$  where  $u$  and  $a$  are the initial velocity in m/sec (= 0) and acceleration in m/sec<sup>2</sup> (= 9.8 m/s<sup>2</sup>)).**

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int u, h, f, distance ;
float a = 9.8f ; // acceleration is constant valuedouble t,
x ;
printf("Enter the number of floors in the building A : ") ;
scanf(" %d ", &f) ;
printf("Enter the height of each floor in the building A : ") ;
scanf(" %d ", &h) ;
distance = f * h ; // number of floor * height of each floor gives distance
u = 0 ; // since
the ball is dropped, and initially it was at rest
x = (2 * distance)/ a ;
t =sqrt(x);
printf(" Time taken : %lf ", t);
getch() ;
}
```



## Output

Since the ball is dropped from the top, the initial velocity is 0.

Given that we have 10 floors each of 3m, the height travelled by the ball is :

$$3 \times 10 = 30 \text{ m}$$

From the equation above :

$$S = ut + \frac{1}{2}gt^2$$

$u$  = the initial velocity = 0

$S$  = height covered = 30m

$g$  = 9.8

Doing the substitution :

$$30 = 0 \times t + 0.5 \times 9.8 \times t^2$$

$$30 = 4.9t^2$$

$$t^2 = 30/4.9 = 6.1224$$

$$t = \sqrt{6.1224}$$

$$t = 2.47 \text{ seconds}$$

**2. Write a program that asks the user to enter the highest rainfall ever in one season for a country, and the rainfall in the current year for that country, obtains the values from the user, checks if the current rainfall exceed the highest rainfall and prints an appropriate message on the screen. If the current rainfall is higher, it assigns that value as the highest rainfall ever. Use only the single-selection form of the if statement.**

```
#include<iostream>
#include<stdio.h>
using namespace std;
int main ()
{
const int NumofMonths = 12;
int rain;
int values[NumofMonths];
const int STRING_SIZE=15;
double avg,total = 0;
int largest,lmonth,smallest,smonth,count;
char p_months[NumofMonths][STRING_SIZE]={"January","February","March","April","May","June",
"July","August", "September", "October","November", "December" };
for (int month = 1; month <= NumofMonths; month++)
{
printf("Enter the total rainfall for ", p_months[month-1]);
scanf("%d",&rain);
if (rain < 0)
{
printf("Do not accept negative numbers Renter");
printf("Please Reenter: ");
```

```

scanf("%d",&rain);
}
values[month-1]=rain;total += rain;
}
printf("The total rainfall for the year is %lf: ");
//scanf("%ld",&total);
avg = total / (double)NumofMonths; printf("The average amount of rainfall is %f ");
for (int month = 1; month <= NumofMonths; month++)
{
largest = values[0];
for (count = 1; count < NumofMonths; count++)
{
if (values[count] > largest)
{
largest = values[count];lmonth=count;
}
}
smallest = values[0];
for (count = 1; count < NumofMonths; count++)
{
if (values[count] < smallest)
{
smallest = values[count];smmonth=count;
}
}
}
printf("The highest month value is : " ,p_months[lmonth]) ;printf("The lowest month value is :
",p_months[smmonth]) ;
//system("pause");
}

```

**Output:**

Enter the rainfall for:

|           |      |
|-----------|------|
| January   | : 22 |
| February  | : 33 |
| March     | : 44 |
| April     | : 56 |
| May       | : 78 |
| June      | : 21 |
| July      | : 90 |
| August    | : 23 |
| September | : 31 |
| October   | : 67 |
| November  | : 70 |
| December  | : 78 |

The total rainfall is 613

The average rainfall is 51.0833 June has the lowest rainfall at 21

July has the highest rainfall at 90

**WEEK-4**  
**EXPERIMENT**

**Program 8:**

**8. Write a C program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.**

**Problem Description:** Prime numbers are positive integers that can only be divided evenly by 1 or themselves. By definition, negative integers, 0, and 1 are not considered prime numbers. The list of the first few prime numbers looks like: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, ...

**Algorithm:**

**Step 1:** Start

**Step 2:** Read the number n

**Step 3:** Initialize flag = 0, i = 2

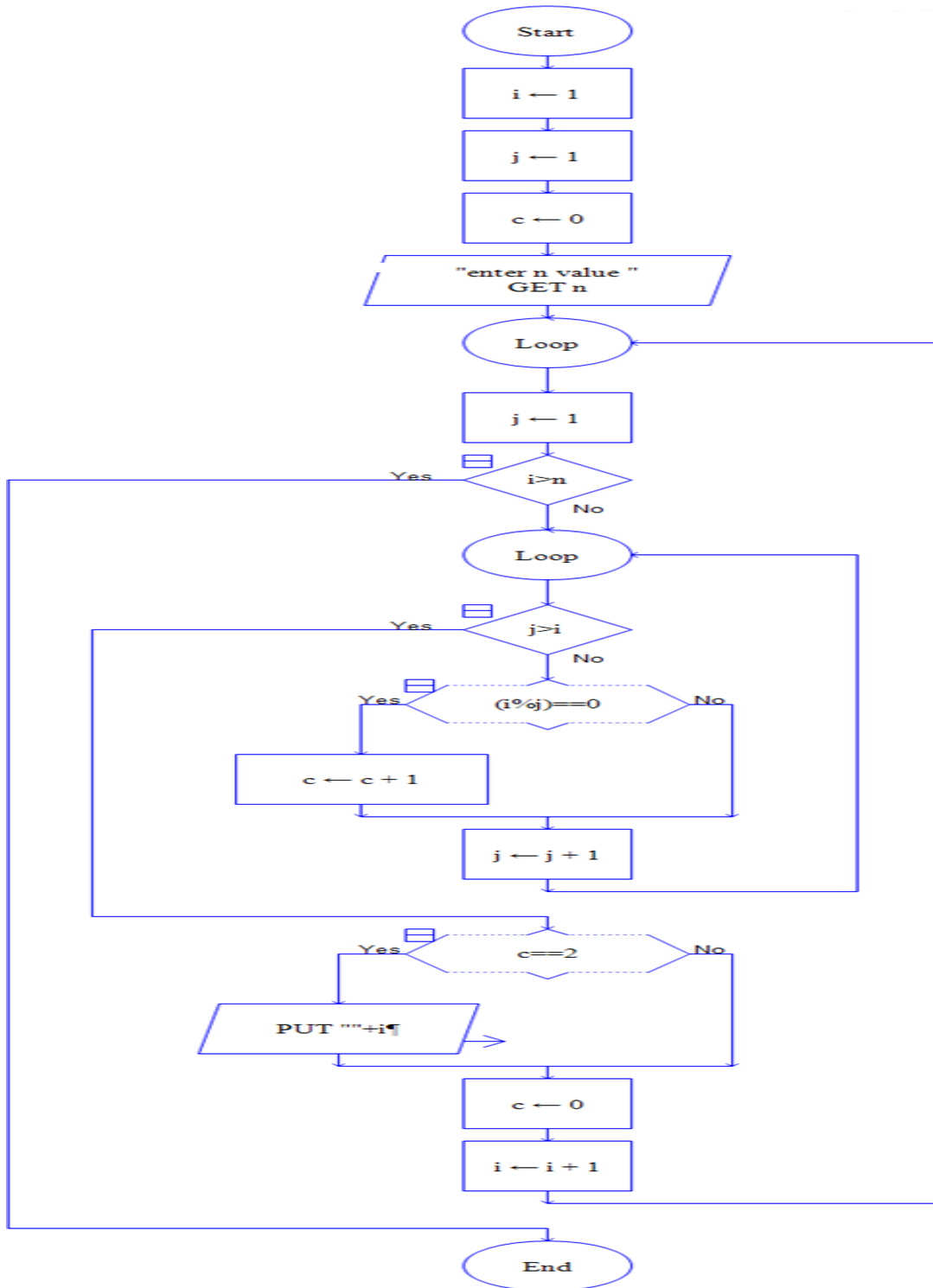
**Step 4:** In for loop check the condition if  $n \% i = 0$  then make  $\text{flag} = \text{flag} + 1$

**Step 5:** If flag = 2 then display the number n as prime number

**Step 6:** else display the number n as not prime number

**Step 7:** Stop

Flow Chart:



## Program 9:

### 9. Write a C program to find the roots of a Quadratic equation

**Problem Description:** This program uses quadratic equation quadratic equation (from the Latin quadratus for "square") is any equation having the form where x represents an unknown, and a, b, and c are constants with a not equal to 0.

The above equation can be solved using the formula

We get two roots from this formula  $x_1$ ,  $x_2$ . and we assign  $b^2-4ac$  to dFirst

the value of d is found. Then the conditions are checked

If  $d > 0$  the roots will be distinctIf

$d = 0$  the roots will be equal $d < 0$

the roots will be complex

#### Algorithm:

**Step 1:** start

**Step 2:** read a,b,c

**Step 3:** compute  $d = b^2 - 4 * a * c$

**Step 4:** if  $d > 0$  then

(4.1) 4.1.1:  $root1 = (-b + \sqrt{d}) / (2 * a)$

4.1.2:  $root2 = (-b - \sqrt{d}) / (2 * a)$

4.1.3: print root1, root2 4.1.4: print

roots are real

else

(4.2) if  $d = 0$  then

4.2.1:  $root1 = -b / (2 * a)$

4.2.2:  $root2 = -b / (2 * a)$

4.2.3: print root1, root2

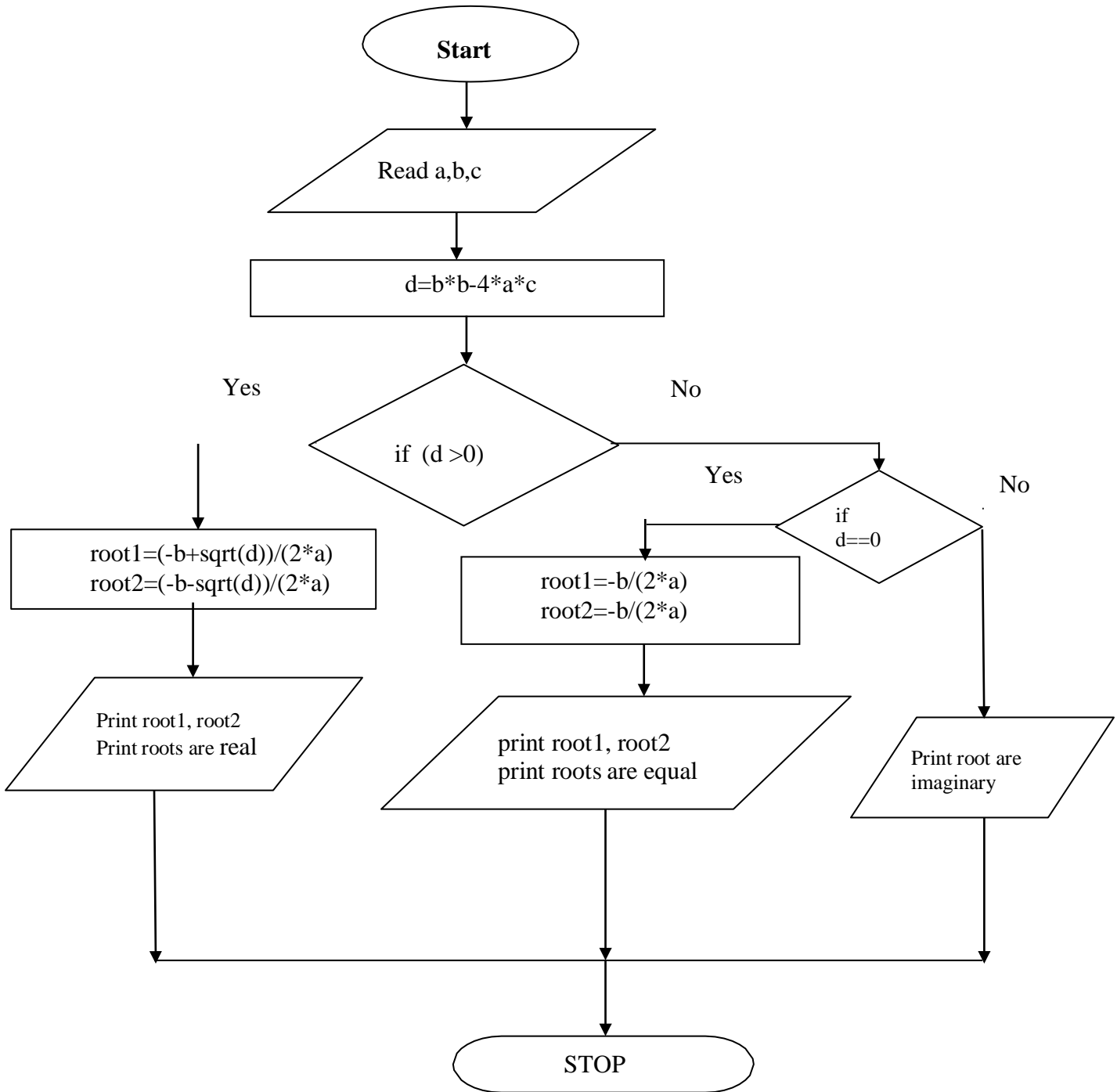
4.2.4: print roots are equal

(4.3) if  $d < 0$

4.3.1: print roots are imaginary

**Step 5:** stop

**Flow chart:**





## VIVA-QUESTIONS:

1. Which of the following cannot be checked in a **switch-case** statement?

- A. Character
- B. Integer
- C. **Float**
- D. enum

2. What is the output?

```
void main()
{
 int x = 5; if (true);
 printf("hello");
}
```

- (a) **Hello**
- (b) Error
- (c) Blank
- (d) None of these

3. What is the output?

```
void main()
{
 int x = 0;
 if (x == 0)
 printf("hi"); else
 printf("how are u");printf("hello");
}
```

- A. hi
- B. how are you
- C. Hello
- D. **hihello**

## Week 5: Iterative Functions

### Demonstration

1. Write a program that reads an integer (5 digits or fewer) and determines and prints how many digits in the integer are 9s.

```
#include <stdio.h>
#include <math.h> /* Used for log10() */
int main()
{
 long long num;
 int count = 0;

 /* Input number from user */
 printf("Enter any number: ");
 scanf("%lld", &num);

 /* Calculate total digits */
 count = (num == 0) ? 1 : (log10(num) + 1);
 printf("Total digits: %d", count);
 return 0;
}
```

#### Output:

```
Enter any number: 123456789Total
digits: 9
```

**2. Write a program that keeps printing the powers of the integer 3, namely 3, 9, 27, 91, 273, and so on. Your loop should not terminate (i.e., you should create an infinite loop). What happens when you run this program.**

```
#include<stdio.h>
main()
{
 //variables
 int term = 1, multiplier = 3, terminus = 1000;
 // terminus value was picked at random
 // Statements
 do
 {
 term = term * multiplier;
 printf("%d\n", term);
 }
 while(term < terminus);
}
```

**Output:**

```
3
9
27
81
243
729
2187
```

**3. Write a program that reads the radius of a circle (as a float value) and computes and prints the diameter, the circumference and the area. Use the value 3.14159 for  $\pi$ .**

```
#include<stdio.h>
int main()
{
float pi = 3.14159, radius, diameter, circumference, area;
printf("Enter the radius of the circle:\n");
scanf("%f", &radius);
 diameter = radius * 2;
printf("The diameter is %f\n", diameter);
 circumference = 2 * pi * radius;
printf("The circumference is %f\n", circumference);
 area = pi * (radius * radius);
printf("The area is %f\n", area);
return 0;
}
```

### **Output**

Enter radius: 10

Diameter = 20 units

**Circumference = 62.79 units Area = 314 sq. units**

## Experiment

**4. Write a menu driven C program that allows a user to enter n numbers and then choose between finding the smallest, largest, sum, or average. The menu and all the choices are to be functions. Use a switch statement to determine what action to take. Display an error message if an invalid choice is entered**

```
#include<stdio.h>
#include<stdlib.h>
#define firstChoice 1
#define secondChoice 2
#define thirdChoice 3
#define fourthChoice 4
/* The program allows a user to enter five numbers and then asks the user to select achoice from a menu.
The menu should offer four options. Use a switch function.
*/ main()
{
double number, smallestNumber, largestNumber, sum, average;
int count = 0;
int choice = 0;
smallestNumber = number;
largestNumber = number;
do {
printf("Enter five numbers. Type -1 to stop. \n");
scanf("%lf", &number);
}
while (number != -1);
printf("\nChoose a selection 1-4 from the menu:\n");
printf("%i. Display the smallest number entered. \n", firstChoice);
```

```

printf("%i. Display the largest number entered. \n", secondChoice);
printf("%i. Display the sum of the five numbers entered. \n", thirdChoice);
printf("%i. Display the average of the five numbers entered. \n", fourthChoice);
scanf("%i", &choice);
switch(choice)
{
case 1:
if(number>smallestNumber&& number>largestNumber)largestNumber
= number;
else
if(number<smallestNumber)
smallestNumber = number;
printf("The smallest number is: %lf", smallestNumber);
break;
case 2:
printf("The largest number is: %lf", largestNumber);
break;
case 3:
sum = sum + number;
count = count + 1;
printf("The sum is: %lf", sum);
break;
case 4:
average = (double)sum/count;
printf("The average is: %lf", average);
break;
default:
printf("That is not a valid number. Please try again.");
break;
}
return 0;
}

```

**Output:**

Enter five numbers. Type -1 to stop.56

Enter five numbers. Type -1 to stop.09

Enter five numbers. Type -1 to stop.89

Enter five numbers. Type -1 to stop.12

Enter five numbers. Type -1 to stop.45

Enter five numbers. Type -1 to stop.

-1

Choose a selection 1-4 from the menu:

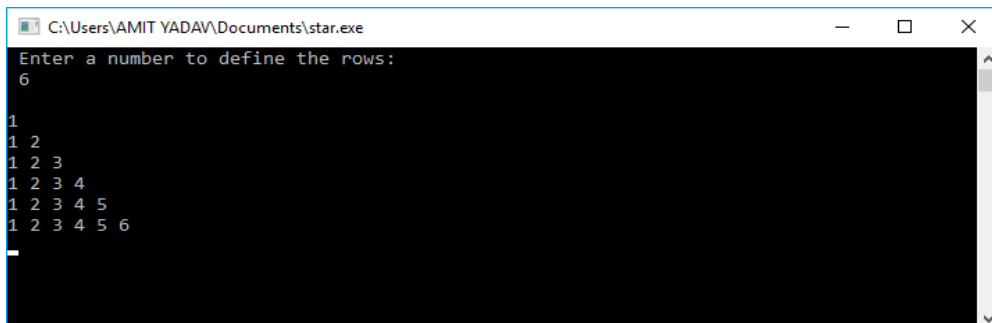
1. Display the smallest number entered.
2. Display the largest number entered.
3. Display the sum of the five numbers entered.
4. Display the average of the five numbers entered.1

**The smallest number is: -1.0000002**

**5. Write a C program to construct a pyramid of numbers as follows:**

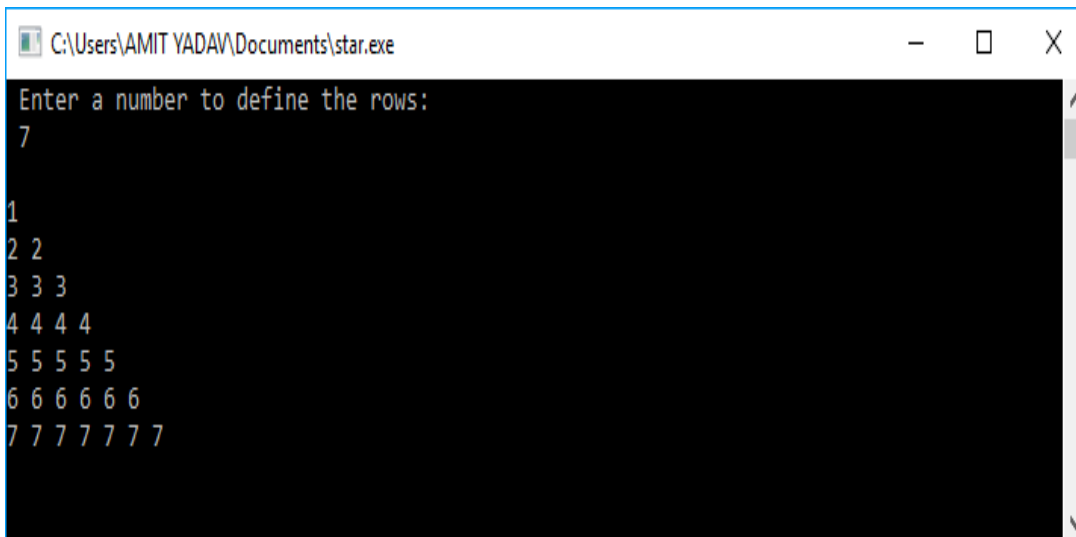
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
#include <stdio.h>
#include <conio.h>
void main ()
{
 // declare the local variables
 int i, j, rows;
 printf (" Enter a number to define the rows: \n ");
 scanf ("%d", &rows);
 printf ("\n");
 for (i = 1; i <= rows; ++i)
 {
 for (j = 1; j <= i; ++j)
 {
 printf ("%d ", j);
 }
 printf ("\n");
 }
 getch();
}
```





```
#include<stdio.h>
#include<conio.h>
void main ()
{
// declare the local variablesint i, j, rows;
printf (" Enter a number to define the rows: \n ");
scanf ("%d", &rows);
printf ("\n");
for (i = 1; i <= rows; ++i)
{
for (j = 1; j <= i; ++j)
{
printf ("%d ", i); // print the number
}
printf ("\n");
}
getch();
}
```



```
C:\Users\AMIT YADAV\Documents\star.exe
Enter a number to define the rows:
7
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
6 6 6 6 6 6
7 7 7 7 7 7 7
```

**Week 7:**  
**Arrays, Pointers and Functions**  
**Demonstration**

**1. Write a C program to find the minimum, maximum and average in an array of integers.**

**Describe:**

The function get result(int arr[],int n) is to find the maximum and minimum element present in the array in minimum no. of comparisons. If there is only one element then we will initialize the variables max and min with arr[0] . For more than one element, we will initialize max with arr[1] and min with arr[0].

**Code:**

```
#include <stdio.h>
int main()
{
int a[8],i,s=0,g,l;
float avg;
printf("Enter 8 Numbers:\n");
for(i=0;i<8;i++)
{
scanf("%d",&a[i]);s=s+a[i];
}
avg=s/8.0;
printf("Sum of Array Elements=%d\n",s);
printf("Average of Elements= %.2f\n",avg);
g=a[0];
for(i=0;i<8;i++) if(a[i]>g)
g=a[i];
printf("Greatest Element= %d\n",g);
l=a[0];
for(i=0;i<8;i++) if(a[i]<l)
l=a[i];
printf("Lowest Element = %d",l);
return 0;
}
```

**Output:**

Enter 8 Numbers:

23

12

45

56

65

78

89

100

Sum of Array Elements = 468

Average of Elements= 58.50

Greatest Element= 100

Lowest Element= 12

## 2. Write a function to compute mean, variance, Standard Deviation, sorting of elements in a single dimension array.

### Describe:

Variance is equal to the average squared deviations from the mean, while standard deviation is the number's square root. Also, the standard deviation is a square root of variance.

### Code:

```
#include<stdio.h>
#include<math.h>
int main ()
{
 int i,n;
 float std_dev,sum=0,sumsq=0,mean,value,variance=0.0,a[100];
 printf("Enter value of n : ");
 scanf("%d",&n); printf("\nEnter numbers : ");
 for(i=0;i<n;i++)
 {
 printf("\nNumber %d : ",i+1);
 scanf("%f",&a[i]);
 sum=sum+a[i];
 }
 mean=sum/n;
 sumsq=0;
 for(i=0;i<n;i++)
 {
 value=a[i]-mean;
 sumsq=sumsq+value*value;
 }
 variance=sumsq/n;
 std_dev=sqrt(variance);
 printf("\nMean of %d numbers = %f\n",n,mean);
 printf("\nVariance of %d numbers = %f\n",n,variance);
```

```
printf("\nStandard deviation of %d numbers = %f\n",n,std_dev);
return 0;
}
```

**Output :**

Enter value of n : 7

Enter numbers :

Number 1 : 23

Number 2 : 45

Number 3 : 12

Number 4 : 66

Number 5 : 58

Number 6 : 31

Number 7 : 67

Mean of 7 numbers = 43.142857

Variance of 7 numbers = 405.551025 Standard  
deviation of 7 numbers = 20.138298

### 3. Write a C program that uses functions to perform the following:

- i. Addition of Two Matrices ii. Multiplication of Two Matrices iii. Transpose of a matrix.

#### i. Addition of Two Matrices

##### Describe:

We can find the sum simply by adding the corresponding entries in matrices A and B.

##### Code:

```
#include<stdio.h>
int rows, columns;
/* adds two matrices and stores the output in third matrix */
void matrixAddition(int mat1[][10], int mat2[][10], int mat3[][10])
{
 int i, j;
 for (i = 0; i < rows; i++) {
 for (j = 0; j < columns; j++) {
 mat3[i][j] = mat1[i][j] + mat2[i][j];
 }
 }
 return;
}
int main() {
 int matrix1[10][10], matrix2[10][10];
 int matrix3[10][10], i, j;
 /* get the number of rows and columns from user */
 printf("Enter the no of rows and columns(<=10):");
 scanf("%d%d", &rows, &columns);
 if (rows > 10 || columns > 10) {
 printf("No of rows/columns is greater than 10\n");
 return 0;
 }
}
```

```

/* input first matrix */
printf("Enter the input for first matrix:");
for (i = 0; i < rows; i++) {
 for (j = 0; j < columns; j++)
 {
 scanf("%d", &matrix1[i][j]);
 }
}
/* input second matrix */
printf("Enter the input for second matrix:");
for (i = 0; i < rows; i++) {
 for (j = 0; j < columns; j++)
 {
 scanf("%d", &matrix2[i][j]);
 }
}
/* matrix addition */ matrixAddition(matrix1, matrix2, matrix3);
/* print the results */
printf("\nResult of Matrix Addition:\n");
for (i = 0; i < rows; i++) {
 for (j = 0; j < columns; j++)
 { printf("%5d", matrix3[i][j]);
 }
 printf("\n");
}
return 0;
}

```

**Output :**

Enter the no of rows and columns(<=10): 3

3

Enter the input for first matrix:10 20

30

40 54 60

70 80 90

Enter the input for second matrix:100

110 120

130 140 150

160 170 180

Result of Matrix Addition:

110 130 150

170 194 210

230 250 270



## ii. Multiplication of Two Matrices

### Describe:

The product of two matrices A and B is defined if the number of columns of A is equal to the number of rows of B. If both A and B are square matrices of the same order, then both AB and BA are defined. If AB and BA are both defined, it is not necessary that  $AB = BA$ .

### Code:

```
#include <stdio.h>

void take_data(int a[][10], int b[][10], int r1, int c1, int r2, int c2);

void multiplication(int a[][10], int b[][10], int mult[][10], int r1, int c1, int r2, int c2);

void display(int mult[][10], int r1, int c2);

int main()
{
 int a[10][10], b[10][10], mult[10][10], r1, c1, r2, c2, i, j, k;
 printf("Enter rows and column for first matrix: ");
 scanf("%d %d", &r1, &c1);
 printf("Enter rows and column for second matrix: ");
 scanf("%d %d", &r2, &c2);
 //Checking if matrix multiplication is possible while
 (c1 != r2)
 {
 printf("\nMatrices with entered orders can't be multiplied with each other.");
 printf("\nMake the column of the first matrix equal to the row of the second.\n");
 printf("\nEnter rows and column for first matrix: ");
 scanf("%d %d", &r1, &c1);
 printf("Enter rows and column for second matrix: ");
 scanf("%d %d", &r2, &c2);
 }
 take_data(a, b, r1, c1, r2, c2);
 multiplication(a, b, mult, r1, c1, r2, c2);
 display(mult, r1, c2);
 return 0;
}
```

```

//This matrix takes the data of matrices.
void take_data(int a[][10], int b[][10], int r1,int c1, int r2, int c2)
{
int i,j;
printf("\nEnter elements of matrix 1:\n");
for(i=0; i<r1; ++i)
for(j=0; j<c1; ++j)
 {
printf("Enter elements a%d%d: ",i+1,j+1);
scanf("%d",&a[i][j]);
 }
printf("\nEnter elements of matrix 2:\n");
for(i=0; i<r2; ++i)
for(j=0; j<c2; ++j)
 {
printf("Enter elements b%d%d: ",i+1,j+1);
scanf("%d",&b[i][j]);
 }
}

//This function multiplies the entered matrices.
void multiplication(int a[][10],int b[][10],int mult[][10],int r1,int c1,int r2,int c2)
{
int i,j,k;
 /* Initializing elements of matrix mult to 0.*/
 for(i=0; i<r1; ++i)
for(j=0; j<c2; ++j)
 {
mult[i][j]=0;
 }
 /* Multiplying matrix a and b and storing in array mult. */
 for(i=0; i<r1; ++i)
for(j=0; j<c2; ++j)

```

```

for(k=0; k<c1; ++k)
{
mult[i][j]+=a[i][k]*b[k][j];
}
}
//This function displays the final matrix after multiplication.
void display(int mult[][10], int r1, int c2)
{
int i, j;
printf("\nThe product of the entered matrices is:\n");
for(i=0; i<r1; ++i)
for(j=0; j<c2; ++j)
{
printf("%d ",mult[i][j]);if(j==c2-1)
printf("\n\n");
}
}

```

**Output:**

Enter rows and column for first matrix: 23

Enter rows and column for second matrix: 32

Enter elements of matrix 1:

Enter elements a11: 3

Enter elements a12: -2

Enter elements a13: 5

Enter elements a21: 3

Enter elements a22: 0

Enter elements a23: 4

Enter elements of matrix 2:

Enter elements b11: 2

Enter elements b12: 3

Enter elements b21: -9

Enter elements b22: 0

Enter elements b31: 0

Enter elements b32: 4

Output Matrix:

24 29

6 25

### iii. Transpose of a matrix with memory

dynamically allocated for the new matrix as row and column counts may not be the same.

#### Describe:

The transpose of a matrix is found by interchanging its rows into columns or columns into rows. The transpose of the matrix is denoted by using the letter "T" in the superscript of the given matrix. For example, if "A" is the given matrix, then the transpose of the matrix is represented by A' or A<sup>T</sup>.

#### Code:

```
#include<stdio.h>

int main()
{
 int a[10][10], transpose[10][10], r, c;
 printf("Enter rows and columns: ");
 scanf("%d %d", &r, &c);
 // assigning elements to the matrix
 printf("\nEnter matrix elements:\n");
 for (int i = 0; i < r; ++i)
 for (int j = 0; j < c; ++j)
 {
 printf("Enter element a%d%d: ", i + 1, j + 1);
 scanf("%d", &a[i][j]);
 }
 // printing the matrix a[][]
 printf("\nEnter matrix: \n");
 for (int i = 0; i < r; ++i)
 for (int j = 0; j < c; ++j)
 {
 printf("%d ", a[i][j]);
 if (j == c - 1)
 printf("\n");
 }
}
```

```

// computing the transpose
for (int i = 0; i < r; ++i)
for (int j = 0; j < c; ++j)
{
 transpose[j][i] = a[i][j];
}

// printing the transpose
printf("\nTranspose of the matrix:\n");
for (int i = 0; i < c; ++i)
for (int j = 0; j < r; ++j)
{
 printf("%d ", transpose[i][j]);
 if (j == r - 1)
 printf("\n");
}
return 0;
}

```

**Output:**

Enter the number of rows and column: 3 3

Enter the elements of the matrix: 1 4 9 7 8 5 2 9 8 The elements in the matrix are:

1 4 9

7 8 5

2 9 8

After transpose the elements are...1 7 2

4 8 9

9 5 8

**WEEK-8**  
**EXPERIMENT**

**Program 6:**

**6. Write a C program to find the GCD (greatest common divisor) of two given integers.**

**Problem Description:**

The GCD (greatest common divisor) of two given integers.

**Algorithm :**

**Step 1:** Start

**Step 2:** Declare variable n1, n2, gcd=1, i=1

**Step 3:** Input n1 and n2

**Step 4:** Repeat until  $i \leq n1$  and  $i \leq n2$

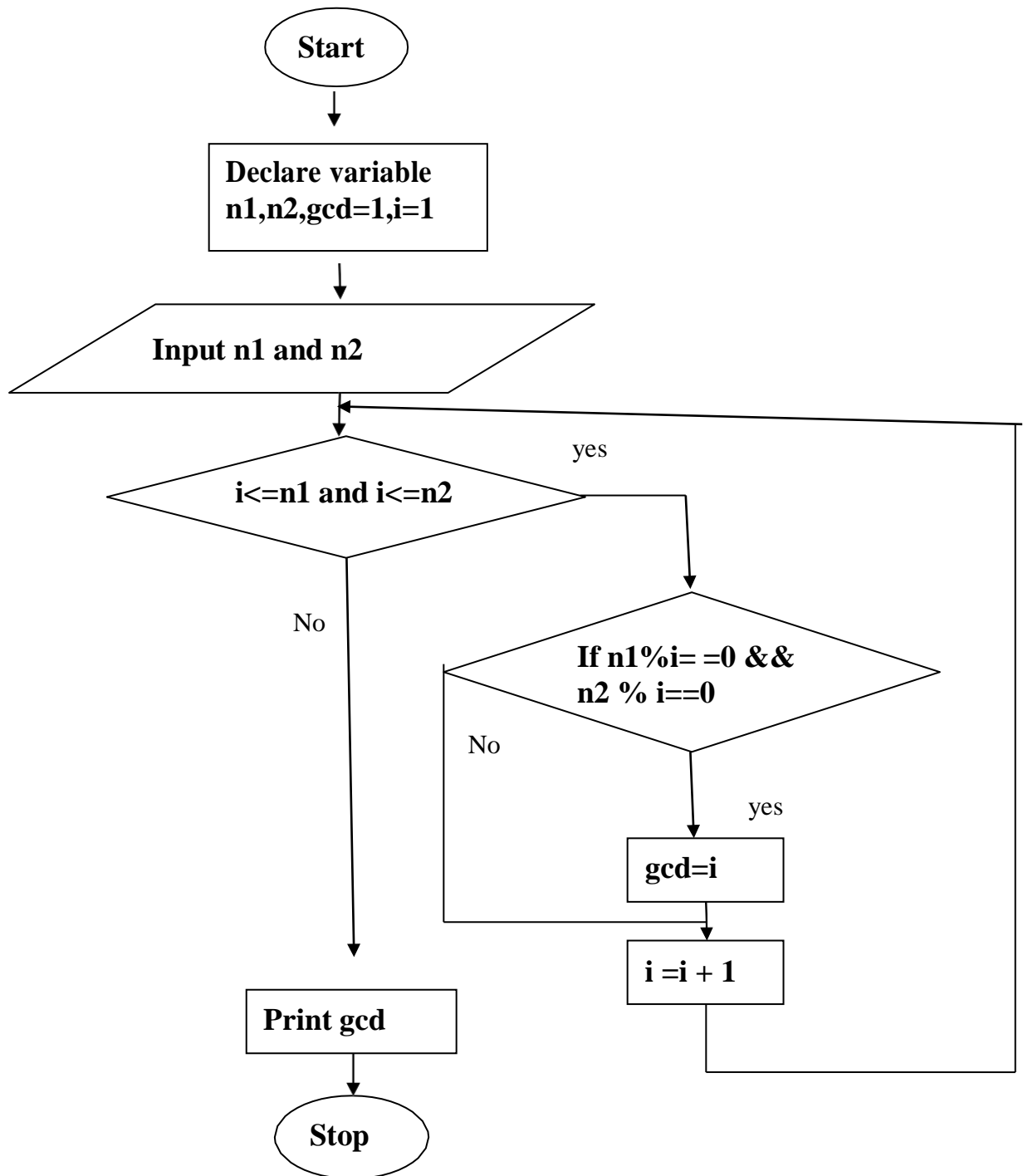
**Step 4.1:** If  $n1 \% i == 0$  &&  $n2 \% i == 0$ :

**Step 4.2:** gcd = i

**Step 5:** Print gcd

**Step 6:** Stop

**Flow chart :**





## 5. Write a C program to compute $x^n$

**Problem Description:** The program uses power function defined in math library. How do you  
If  $n$  is a positive integer and  $x$  is any real number, then  $x^n$  corresponds to repeated multiplication  
 $x^n = x \times x \times \dots \times x$   $n$  times. We can call this “ $x$  raised to the power of  $n$ ,” “ $x$  to the power of  $n$ ,” or simply  
“ $x$  to the  $n$ .” Here,  $x$  is the base and  $n$  is the exponent or the power.

### Algorithm:

**Step 1:** Start

**Step 2:** Declare variable  $pow$  and  $i$ .

**Step 3:** Initialize  $pow = 1$  and  $i = 1$ .

**Step 4:** Read base  $X$  and power  $Y$  from user.

**Step 5:** Repeat step until  $i$  is less than equal to  $Y$  i.e  $i \leq Y$

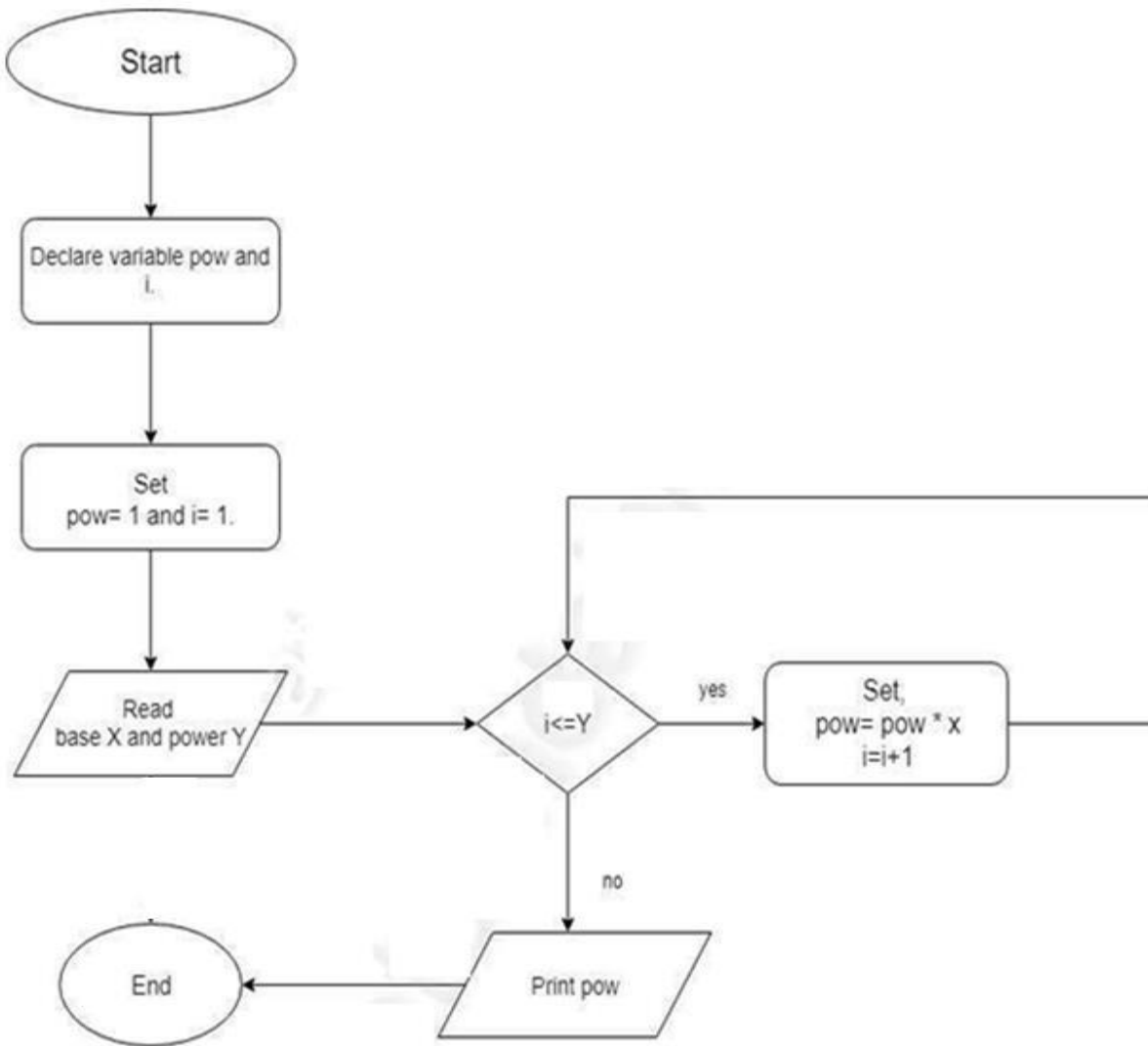
**5.1:** Set,  $pow = pow * x$

**5.2:** Increment the value of  $i$  by 1

**Step 6:** The value stored in  $pow$  is the required value.

**Step 7:** Stop

**Flow chart:**



## VIVA-QUESTIONS:

1. What is the base data type of a pointer variable by which the memory would be allocated to it?

- A. int
- B. float
- C. No datatype
- D. **unsigned int**

2. Can you combine the following two statements into one? `char *p; p = (char*) malloc(100);`

- A. **`char *p = (char*)malloc(100);`**
- B. `char *p = (char*)(malloc*)(100);`
- C. `char *p = (char) malloc(100);`
- D. `char p = *malloc(100);`

3. The keyword used to transfer control from a function back to the calling function is

- A. goto
- B. **return**
- C. switch
- D. exit

4. What is the default return type if it is not specified in function definition?

- A. **int**
- B. short int
- C. void
- D. float

5. A function which calls itself is called a \_\_\_\_\_ function.

- A. **Recursive Function**
- B. Static Function
- C. Self Function
- D. Auto Function

6. In C, if you pass an array as an argument to a function, what actually gets passed?

- A. Value of elements in array
- B. First element of the array
- C. Base address of the array
- D. Address of the last element of array

7. The parameter passing mechanism for an array is

- A. call by reference
- B. call by value
- C. call by value-result
- D. None of the above

**Week 11:**  
**VI Strings**  
**Demonstration**

**1. Write a C program to convert a Roman numeral ranging from I to L to its decimal equivalent.**

**Describe:**

Roman numerals are represented by seven different symbols: I, V, X, & L

| Symbol | Value |
|--------|-------|
| I      | 1     |
| V      | 5     |
| X      | 10    |
| L      | 50    |

**Code:**

```
#include<stdio.h>
//#include<conio.h>
#include<string.h>
//Declaration of function decimal int
decimal(char);
void main()
{
char roman_Number[1000];
int i=0;
long int number =0;
//clrscr();
printf("\nEnter any roman number (Valid digits are I, V, X, L): \n");
scanf("%s",roman_Number);
while(roman_Number[i])
{
```

```

if(decimal(roman_Number[i]) < 0)
{
printf("Invalid roman digit : %c",roman_Number[i]);
}
if((strlen(roman_Number) -i) > 2)
{
if(decimal(roman_Number[i])<decimal(roman_Number[i+2]))
{
printf("Invalid roman number");
}
}
if(decimal(roman_Number[i])>=decimal(roman_Number[i+1]))
number = number + decimal(roman_Number[i]);
else
{
number = number + (decimal(roman_Number[i+1]) - decimal(roman_Number[i]));
i++;
}
i++;
}
printf("Its decimal value is : %ld",number);
//getch();
}
//Definition of function decimal int
decimal(char ch)
{
int value=0;
//switch statement to assign values to the decimal digits of roman numerals

```

```
switch(ch)
{
case 'I': value = 1;
break;
case 'V': value = 5;
break;
case 'X': value = 10;
break;
case 'L': value = 50;
break;
case '\0': value = 0;
break;
default: value = -1;
}
return value;
}
```

**Output :**

Enter any roman number (Valid digits are I, V, X, L):

XXX

Its decimal value is: 30

## 2. Write a C program that converts a number ranging from 1 to 50 to Roman equivalent c.

### Describe:

In the Roman numeral system, the symbols I, V, X, and L stand respectively for 1, 5, 10, and 50 in the Hindu-Arabic numeral system.

### Code:

```
#include<stdio.h>
int main()
{
int n;
printf("Decimal Roman\n");
printf("numbers numerals\n");
printf("- _____-\n");
for(int i=1; i<=50; i++)
{
n = i; printf("%d",i);
while(n != 0)
{
if (n >= 100)
{
printf("C");n -= 100;
}
else if (n >= 50)
{
printf("L");n -= 50;
}
else if (n >= 40)
{
printf("XL");n -= 40;
}
else if (n >= 10)
{
```



```
printf("X");n -= 10;
}
else if (n >= 9)
{
printf("IX");n -= 9;
}
else if (n >= 5)
{
printf("V");n -= 5;
}
else if (n >= 4)
{
printf("IV");n -= 4;
}
else if (n >= 1)
{
printf("I");n -= 1;
}
}
printf("\n");
}
return 0;
}
```

**Output:**

Decimal Roman numbers numerals

-----

1I

2II

3III

4IV

5V

6VI

7VII

8VIII

9IX

10X

11XI

12XII

13XIII

14XIV

15XV

16XVI

17XVII

18XVIII

19XIX

20XX

21XXI

22XXII

23XXIII

24XXIV

25XXV

26XXVI

27XXVII

28XXVIII

29XXIX

30XXX

31XXXI

32XXXII

33XXXIII

34XXXIV

35XXXV

36XXXVI  
37XXXVII  
38XXXVIII  
39XXXIX  
40XL  
41XLI  
42XLII  
43XLIII  
44XLIV  
45XLV  
46XLVI  
47XLVII  
48XLVIII  
49XLIX  
50L

### 3. Write a C program that uses functions to perform the following operations:

- **To insert a sub-string into a given main string from a given position. Describe:**

Insert the substring from 0 to the specified (index + 1) using substring (0, index+1) method. Then insert the string to be inserted into the string. Then insert the remaining part of the original string into the new string using substring(index+1) method. Return/Print the new String.

#### **Code:**

```
#include<stdio.h>
#include<string.h>
#include <stdlib.h>
Void insert_substring(char*, char*, int);
char* substring(char*, int, int);
int main()
{
char text[100], substring[100];
int position;
printf("Enter some text\n");
gets(text);
printf("Enter a string to insert\n");
gets(substring);
printf("Enter the position to insert\n");
scanf("%d", &position);
insert_substring(text, substring, position);
printf("%s\n",text);
return 0;
}
Void insert_substring(char *a, char *b, int position)
{
char *f, *e;int length;
length = strlen(a);
f = substring(a, 1, position - 1);
e = substring(a, position, length-position+1);
strcpy(a, "");
```

```

strcat(a, f);
free(f);
strcat(a, b);
strcat(a, e);
free(e);
}
char *substring(char *string, int position, int length)
{
char *pointer;int c;
pointer = malloc(length+1);
if(pointer==NULL) exit(EXIT_FAILURE);
for(c = 0 ; c < length ; c++)
 *(pointer+c) = *((string+position-1)+c);
*(pointer+c) = '\0';
return pointer;
}

```

**Output:**

Enter some text computer is amazing  
Enter a string to insertprogramming  
Enter the position to insert 10  
Computer programming is amazing

- **To delete n Characters from a given position in a given string.**

**Describe:**

C Program to delete the n characters from a given position from a given string. Here we use the gets() and puts() functions to read and write the string. delchar() function reads the character string and checks the length and position, then using strcpy() function it replaces the original string.

**Code:**

```
#include<stdio.h>
#include<conio.h>
#include <string.h>
void delchar(char *x,int a, int b);
void main()
{
 char string[10];
 int n,pos,p;
 puts("Enter the string");
 gets(string);
 printf("Enter the position from where to delete");
 scanf("%d",&pos);
 printf("Enter the number of characters to be deleted");
 scanf("%d",&n);
 delchar(string,n,pos);getch();
}
void delchar(char *x,int a, int b)
{
 if ((a+b-1) <= strlen(x))
 {
 strcpy(&x[b-1],&x[a+b-1]);
 puts(x);
 }
}
```

**Output :**

Enter the string computer is good

Enter the position from where to delete 5

Enter the number of characters to be deleted 8

Comp good

**WEEK-12**  
**EXPERIMENT**

**Program 4:**

**4. Write a C program to determine if the given string is a palindrome or not (Spelled same in both directions with or without a meaning like madam, civic, noon, abcba, etc.)**

**Problem Description:**

1. Take a string as input.
2. Store the string in the stack array.
3. Check whether the string is palindrome or not.

**Algorithm :**

**Step 1.:**Start

**Step 2.:**Read the string from the user

**Step 3.:** Calculate the length of the string

**Step 4.:** Initialize rev = ""[empty string]

**Step 5.:**Initialize i = length - 1

**Step 6.:**Repeat until i >= 0:

**6.1:** rev = rev + Character at position 'i' of the string

**6.2:** i = i - 1

**Step 7.:**If string = rev:

**7.1:** Print "Given string is palindrome"

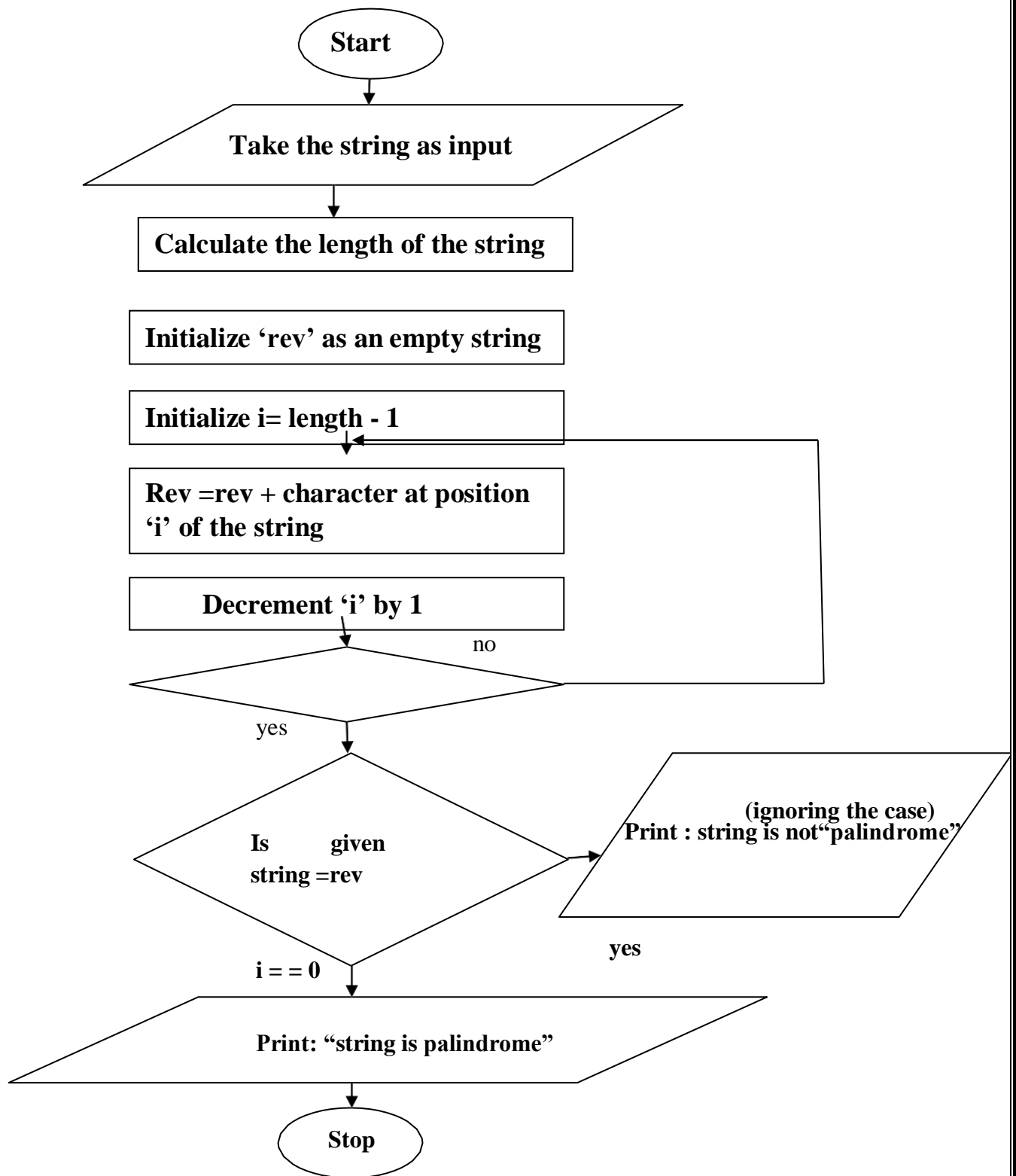
**Step 8.:** Else:

**8.1 :** Print "Given string is not palindrome"

**Step 9.:**Stop

**Flow chart :**





**Program 5:**

**5. Write a C program that displays the position of a character ch in the string S or - 1 if S doesn't contain ch.**

**Problem Description:**

This C Program checks whether a given character is present in a string, it also displays its occurrence and frequency.

**Algorithm :**

**Step 1:** Start

**Step 2:** read the string and then displayed

**Step 3:** read the string to be searched and then displayed

**Step 4:** searching the string T in string S and then perform the following steps

**i.** found = strstr(S, T)

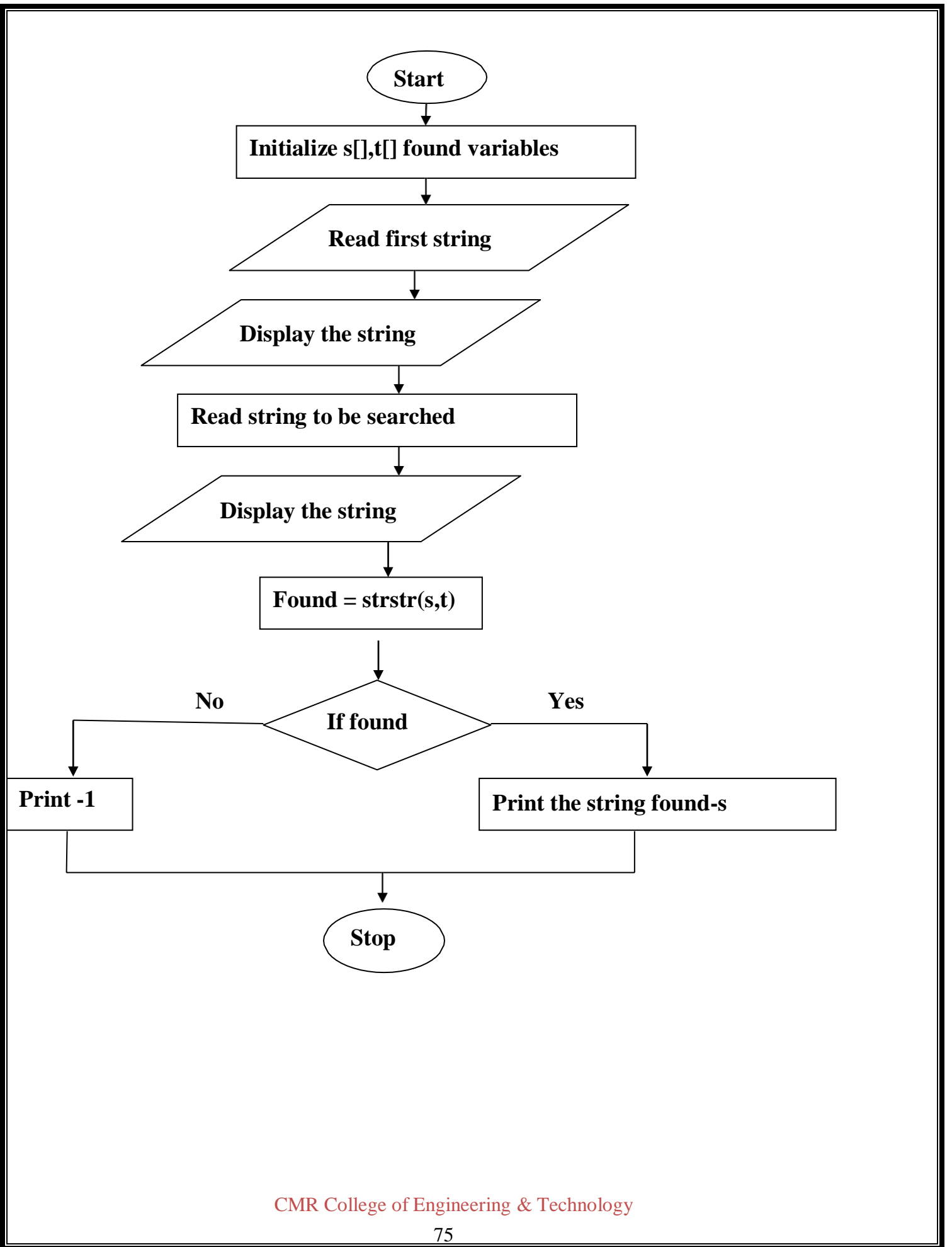
**ii.** if found print the second string is found in the first string at the position.

**iii.** If not goto step5

**Step 5:** print the -1

**Step 6:** Stop

**Flow chart :**



## Program 6:

### 6. Write a C program to count the lines, words and characters in a given text.

#### Problem Description:

1. Take a string as input.
2. Using for loop search for a empty space in between the words in the string.
3. Consecutively increment a variable. This variable gives the count of number of words.

#### Algorithm:

**Step 1:** Take a string as input and store it in the array of characters.

**Step 2:** Create 3 counter variables for the count of words, lines and characters in the string.

**Step 3:** Using for loop search for a space ' ' in the string and consecutively increment the variable count for words.

**Step 4:** Using for loop search for a next line '\n' in the string and consecutively increment a variable count for next line.

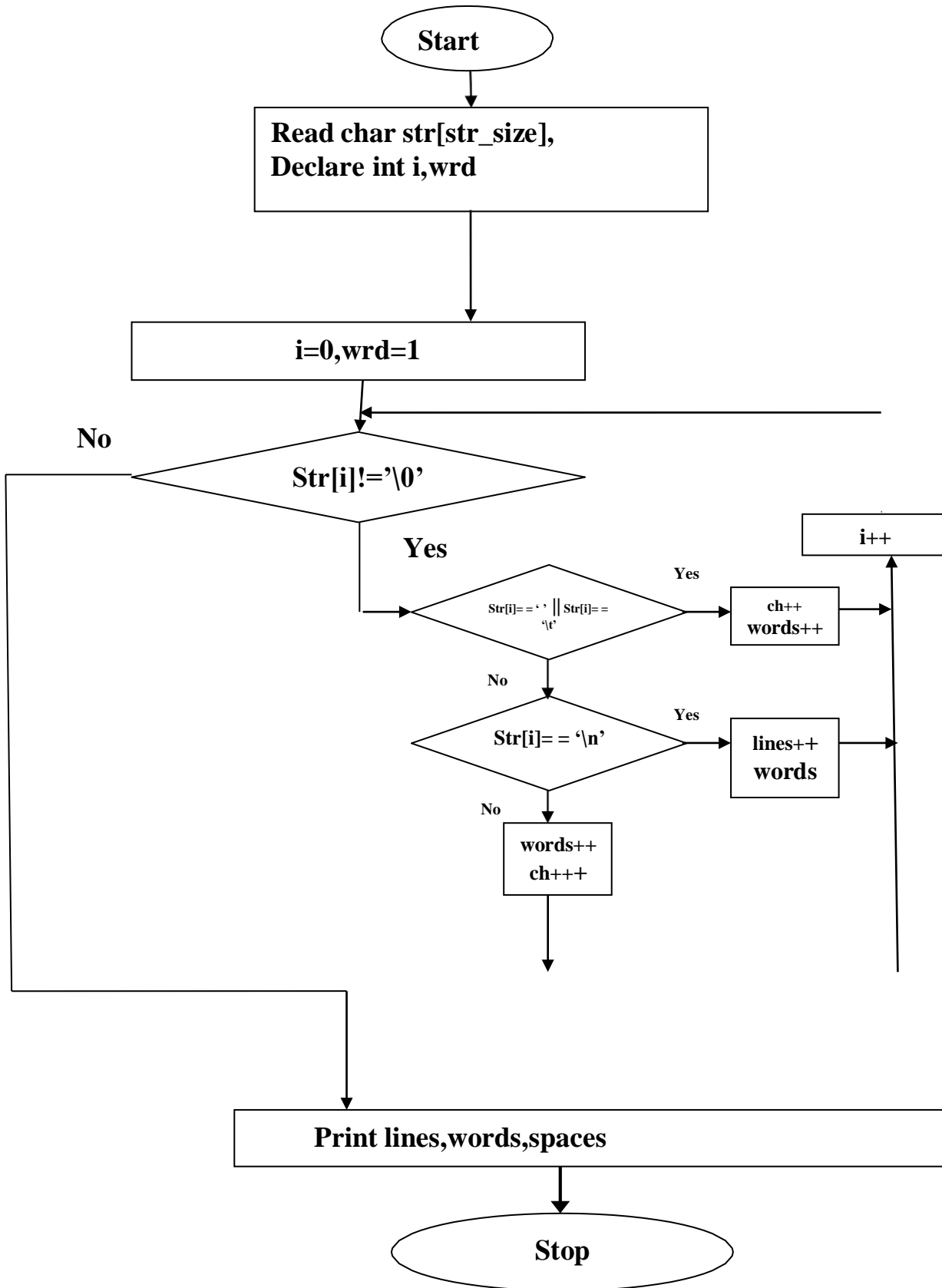
**Step 5:** Using for loop search for characters except space and new line in the string and consecutively increment a variable count for characters.

**Step 6:** Repeat step 3,4,5 until the loop reaches to the end of the string.

**Step 7:** Check for the corner cases and do accordingly.

**Step 8:** Print all the values of the counter.

#### Flow chart :



## VIVA-QUESTIONS:

1. A string in C is

- A. 1-D Array of Character
- B. 2-D Array of Character
- C. Any of A & B
- D. None of the above

2. Which function will you choose to join two words?

- A. strncon()
- B. memcon()
- C. strepy()
- D. strcat()

3. If the two strings are identical, then strcmp() function returns

- A. 0
- B. 1
- C. -1
- D. None of these

4. A String constant in C terminated by

- A. '\0'
- B. '\\0'
- C. "
- D. " "

5. The library function used to find the last occurrence of a character in a string is

- A. strrchr() // It scans a string *s* in the reverse direction, looking for a specific character *c*.
- B. strstr()
- C. strnstr()
- D. laststr()

6. To receive multi-word string from keyboard which of the function is more appropriate?

- A. Scanf
- B. `gets()`
- C. both
- D. None of the above

**Q1. Write a program that uses functions to perform the following operations on singly linked list**

- i) Creation ii) Insertion iii) Deletion iv) Traversal**

Linked List is a linear data structure in which every data item is represented as Node containing two or more lots.

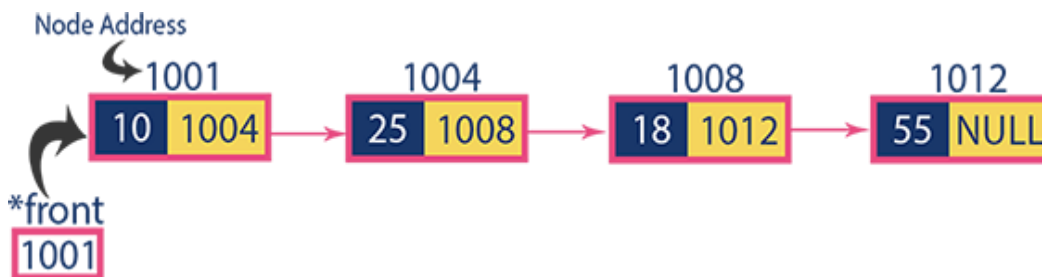
**Single linked list is a sequence of elements in which every element has link to its next element in the sequence.**

In any single linked list, the individual element is called as "Node". Every "Node" contains two fields, data field and next field. The data field is used to store actual value of the node and next field is used to store the address of next node in the sequence.

The graphical representation of node in a single linked list is as follows...



Example



**Operations on Single Linked List**

The following operations are performed on a Single Linked List

- Creation
- Insertion
- Deletion
- Traversal

Before we implement actual operations, first we need to setup empty list. First perform the following steps before implementing actual operations.



## Node Creation

```
struct node
{
 int data;
 struct node *next;
};
struct node *head, *ptr;
ptr = (struct node *) malloc (sizeof(struct node *));
```

**Step1-** Include all the **header files** which are used in the program.

**Step2-** Declare all the **user defined functions**.

**Step3 -** Define a **Node** structure with two members **data** and **next**

**Step4-** Define a Node pointer '**starts**' and set it to **NULL**.

**Step5-** Implement the main method by displaying operations menu and make suitable function calls in the main method to perform user selected operation.

## Insertion:

In a single linked list, the insertion operation can be performed in three ways. They are as follows...

1. Inserting At Beginning of the list
2. Inserting At End of the list
3. Inserting At Specific location in the list

### Inserting At Beginning of the list:

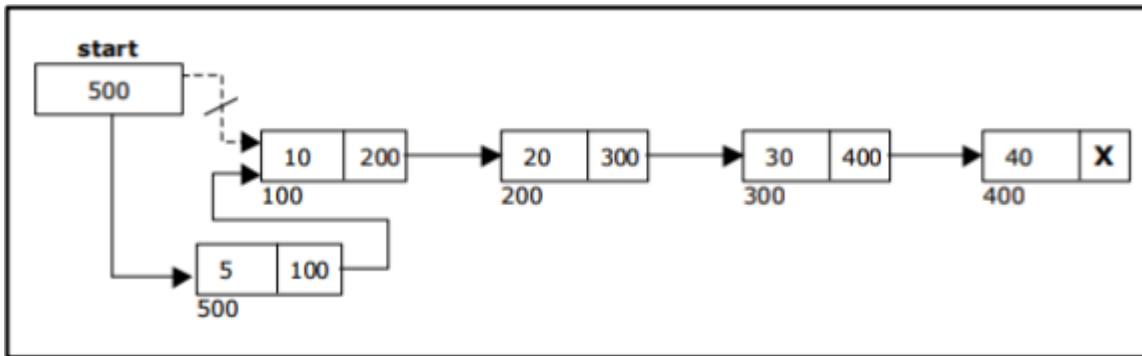
We can use the following steps to insert anew node at beginning of the single linked list...

**Step1-** Create a newNode with given value.

**Step2-** Check whether list is **Empty** (**start==NULL**)

**Step3-** If it is **Empty** then, set newNode→**next** =**NULL** and **start=newNode**.

**Step4-** If it is **NotEmpty** then, set **newNode→next=start** and **start=newNode**.



### Inserting At End of the list:

We can use the following steps to insert a new node at end of the single linked list...

**Step1-**Create a **new Node** with given value and **newNode**→**next** as **NULL**.

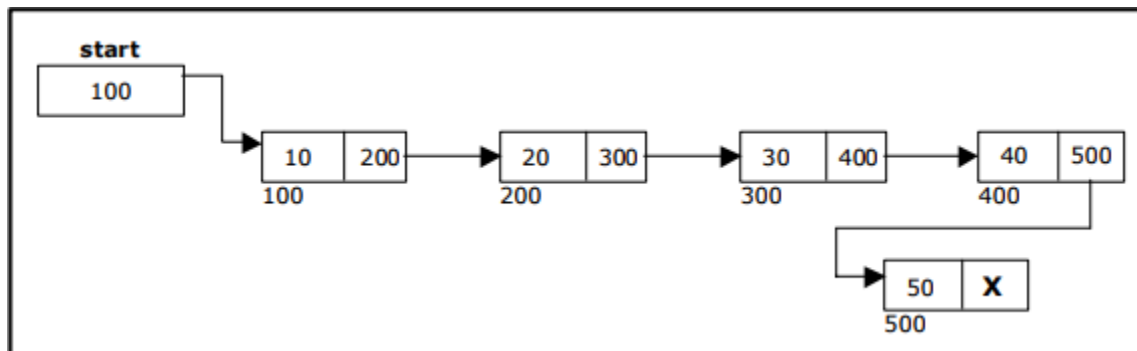
**Step2-**Check whether list is **Empty** (**start**==**NULL**).

**Step3-**If it is **Empty** then, set **start**=**newNode**.

**Step4-** If it is **NotEmpty** then defines a node pointer **temp** and initialize with **start**.

**Step5-**Keep moving the **temp** to its next node until it reaches to the last node in the list (until **temp** →**next** is not equal to **NULL**).

**Step6-**Settemp → next=newNode.



### Inserting at middle of the list:

**Step1**-Create a new Node with given value

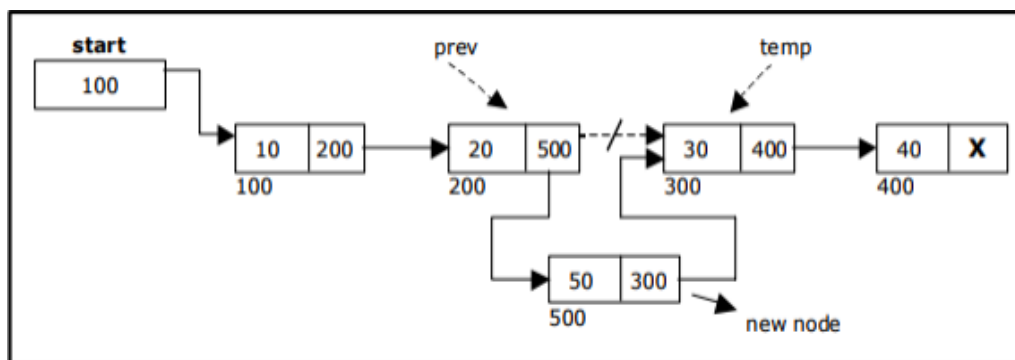
**Step2**-Check whether list is **Empty** (**start==NULL**)

**Step 3** - If it is **Empty** then, set **newNode** → **next = NULL** and **start = newNode**.

**Step4**- If it is **NotEmpty** then defines a node pointer **temp** and initialize with **start**.

**Step5**-Keep moving the **temp** to its next node until it reaches other node after which we want to insert the newNode

**Step6**-Finally, **Setp->next=newnode**, **newnode->next=temp**



### Deletion:

In a singly linked list, the deletion operation can be performed in three ways. They are as follows...

1. Deleting from Beginning of the list
2. Deleting from End of the list
3. Deleting a Specific Node

### Deleting from Beginning of the list:

We can use the following steps to delete a node from beginning of the single linked list...

**Step1-**Check whether list is **Empty** ( $\text{start} == \text{NULL}$ )

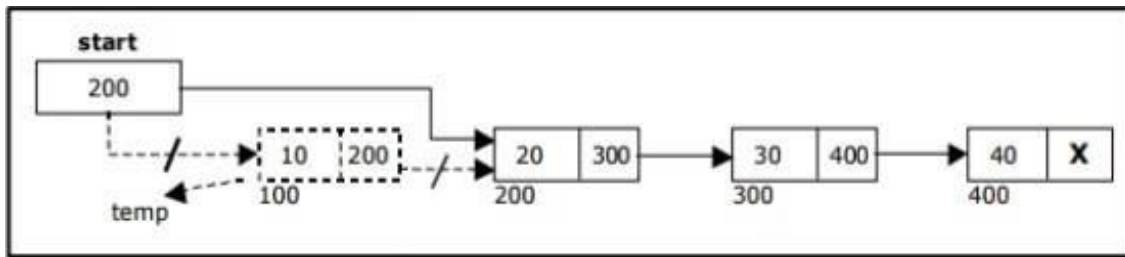
**Step2-**If it is **Empty** then, display 'List is Empty!!!Deletion is not possible' and terminate the function.

**Step3-**If it is **Not Empty** then, define a Node pointer '**temp**' and initialize with **start**.

**Step 4 -** Check whether list is having only one node ( $\text{temp} \rightarrow \text{next} == \text{NULL}$ )

**Step5-** If it is **TRUE** then set  $\text{start} = \text{NULL}$  and delete **temp** (Setting **Empty** list conditions)

**Step6-** If it is **FALSE** then set  $\text{start} = \text{temp} \rightarrow \text{next}$ , and delete **temp** ( $\text{free}(\text{temp})$ ).



### Deleting from End of the list:

We can use the following steps to delete a node from end of the single linked list...

**Step1-**Check whether list is **Empty** ( $\text{start} == \text{NULL}$ )

**Step2-** If it is **Empty** then, display 'List is Empty!!!Deletion is not possible' and terminate the function.

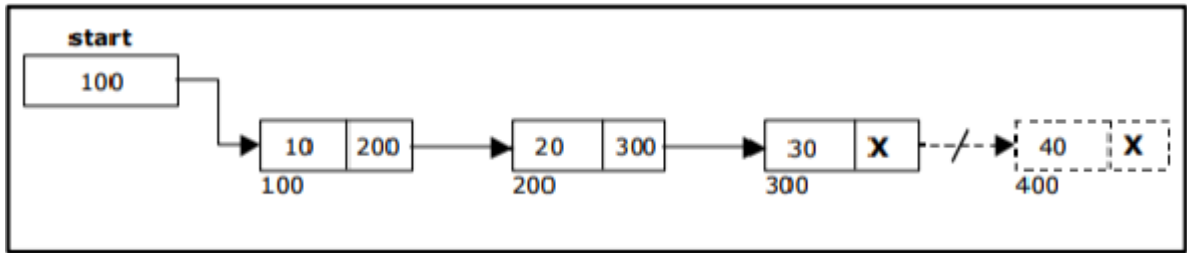
**Step 3 -** If it is **Not Empty** then, define two Node pointers '**temp**' and '**temp1**' and initialize '**temp**' with **start**.

**Step4 -** Check whether list has only one Node ( $\text{temp} \rightarrow \text{next} == \text{NULL}$ )

**Step5-** If it is **TRUE**. Then, set  $\text{start} = \text{NULL}$  and delete **temp**. And terminate the function. (Setting **Empty** list condition)

**Step6-** If it is **FALSE**. Then, set '**temp1=temp**' and move **temp** to its next node. Repeat the same until it reaches to the last node in the list. (until  $\text{temp1} \rightarrow \text{next} == \text{NULL}$ )

**Step7-** Finally, Set  $\text{temp1} \rightarrow \text{next} = \text{NULL}$  and delete **temp** ( $\text{free}(\text{temp})$ ).



### Deleting a Specific Node from the list:

**Step1**-Create a new Node with given value

**Step2**-Check whether list is **Empty**( $start == NULL$ )

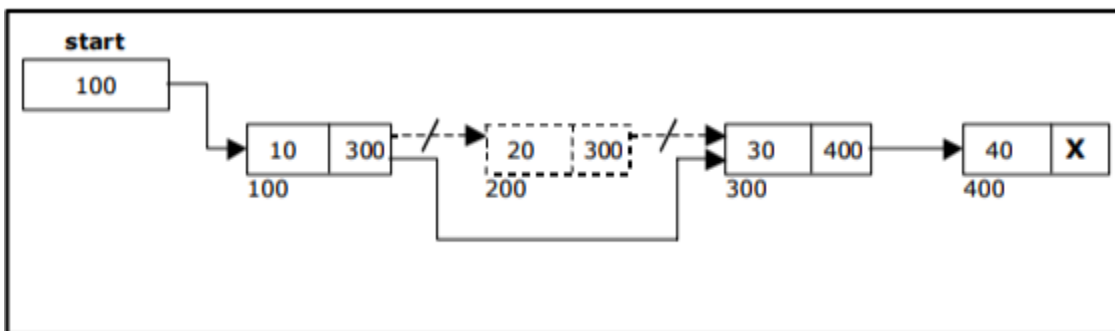
**Step 3** - If it is **Empty** then, set **newNode**  $\rightarrow$  **next** = **NULL** and **start** = **newNode**.

**Step4**- If it is **Not Empty** then defines an order pointer **temp** and initialize with **start**.

**Step5**-Keep moving the **temp** to its next node until it reaches to the node after which we want to delete the Node

**Step6**-Finally, Set  $p \rightarrow next = temp \rightarrow next$

**Step7**-delete emp( $free(temp)$ ).



### Traversing:

- Assign the address of start pointer to at emp pointer.
- Display the information from the data field of each node.
- The function traverse () is used for traversing and displaying the information stored in the list from left to right.

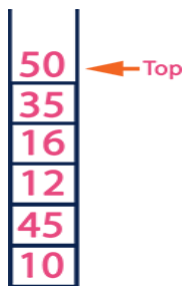
## 2. Write a program that implement stack (its operations) using i) Arrays ii) Pointers

**Stack is a linear data structure.**

"A Collection of similar data items in which both insertion and deletion operations are performed based on LIFO principle".

### Example:

If we want to create a stack by inserting 10, 45,12,16,35 and 50. Then 10 becomes the bottommost element and 50 is the top most element. The last inserted element 50 is at Top of the stack as shown in the image below...



The following operations are performed on the stack...

1. Push (To insert an element on to the stack)
2. Pop (To delete an element from the stack)
3. Display (To display elements of the stack)

Stack data structure can be implemented in two ways. They are as follows...

1. Using Array
2. Using Linked List

When stack is implemented using array, that stack can organize only limited number of elements.

When stack is implemented using linked list, that stack can organize unlimited number of elements.

### Stack Using Arrays:

A stack data structure can be implemented using one dimensional array. But stack implemented using array stores only fixed number of data values. This implementation is very simple. Just define a one-dimensional array of specific size and insert or delete the values into that array by using **LIFO principle** with the help of a variable called '**top**'. Initially top is set to -1. Whenever we want to insert a value into the stack, increment the top value by one and then insert. Whenever we want to delete a value from the stack, then delete the top value and decrement the top value by one.

### **Stack Operations using Arrays:**

A stack can be implemented using array as follows...

Before implementing actual operations, first follow the below steps to create an empty stack.

**Step1**-Include all the **header files** which are used in the program and define a constant **'SIZE'** with specific value.

**Step2**-Declare all the **functions** used in stack implementation.

**Step3**-Create a one-dimensional array with fixed size (**int stack[SIZE]**)

**Step4**-Define an integer variable **'top'** and initialize with **'-1'**. (**int top=-1**)

**Step5**-In main method, display menu with list of operations and make suitable function calls to perform operation selected by the user on the stack.

### **Push (value)-Inserting value into the stack:**

In a stack, **push ()** is a function used to insert an element into the stack. In a stack, the new element is always inserted at **top** position. Push function takes one integer value as parameter and inserts that value into the stack. We can use the following steps to push an element on to the stack...

- **Step 1**-Check whether **stack** is **FULL**. (**top==SIZE-1**)
- **Step2**-If it is **FULL**, then display **"Stack is FULL!!! Insertion is not possible!!!"** and terminate the function.
- **Step3**-If it is **NOT FULL**, then increment **top** value by one (**top++**) and set **stack[top]** to **value(stack[top] =value)**.

### **Pop()-Delete a value from the Stack:**

In a stack, **pop()** is a function used to delete an element from the stack. In a stack, the element is always deleted from **top** position. Pop function does not take any value as parameter. We can use the following steps to pop an element from the stack...

**Step1**-Check whether **stack** is **EMPTY**. (**top==-1**)

**Step 2** - If it is **EMPTY**, then display **"Stack is EMPTY!!! Deletion is not possible!!!"** and terminate the function.

**Step3**-If it is **NOT EMPTY**, then delete **stack[top]** and decrement **top** value by one (**top--**).

### Display ()-Displays the elements of a Stack:

We can use the following steps to display the elements of a stack...

**Step1-Check whether stack is EMPTY.** (top== -1)

**Step2-If it is EMPTY, then display" Stack is EMPTY!!!" and terminate the function.**

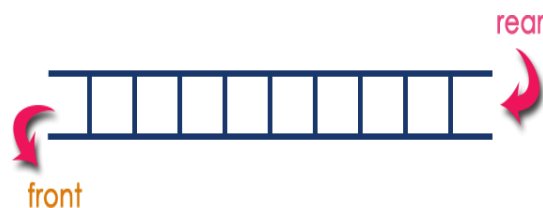
**Step3- If it is NOT EMPTY, then define variable 'I' and initialize with top. Display stack[i] value and decrement i value by one (i--).**

**Step3-Repeat above step until I value becomes '0'.**

### 3. Write a program that implement Queue (its operations) using i) Arrays ii) Pointer

Queue is a linear data structure in which the insertion and deletion operations are performed at two different ends. In a queue data structure, adding and removing of elements are performed at two different positions. The insertion is performed at one end and deletion is performed at other end. In a queue data structure, the insertion operation is performed at a position which is known as '**rear**' and the deletion operation is performed at a position which is known as '**front**'.

In Queue data structure, the insertion and deletion operations are performed based on **FIFO(First In First Out)** principle.



In a queue data structure, the insertion operation is performed using a function called "**enQueue()**" and deletion operation is performed using a function called "**deQueue()**".

Queue after inserting 25,30, 51, 60and 85.

**After Inserting five elements...**





### Operations on a Queue:

The following operations are performed on a queue data structure...

1. enQueue (value)-(To insert an element in to the queue)

deQueue()-(To delete an element from the queue)

3. display()-(To display the elements of the queue)

Queue data structure can be implemented in two ways. They are as follows...

1. Using Arrays
2. Using Pointers

When a queue is implemented using array, that queue can organize only limited number of elements.

When a queue is implemented using linked list, that queue can organize unlimited number of elements.

### Queue Data structure Using Arrays:

A queue data structure can be implemented using one dimensional array. The queue implemented using array stores only fixed number of data values. The implementation of queue data structure using array is very simple. Just define a one dimensional array of specific size and insert or delete the values into that array by using **FIFO (First In First Out) principle** with the help of variables '**front**' and '**rear**'. Initially both '**front**' and '**rear**' are set to -1. Whenever, we want to insert a new value into the queue, increment '**rear**' value by one and then insert at that position. Whenever we want to delete a value from the queue, then delete the element which is at '**front**' position and increment '**front**' value by one.

### Queue Operations using Arrays:

Queue data structure using array can be implemented as follows...

Before we implement actual operations, first follow the below steps to create an empty queue.

**Step1-** Include all the **header files** which are used in the program and define a constant '**SIZE**' with specific value.

**Step2-** Declare all the **user defined functions** which are used in queue implementation.

**Step3-** Create a one dimensional array with above defined SIZE (**int queue[SIZE]**)

**Step4-** Define two integer variables '**front**' and '**rear**' and initialize both with '**-1**'. (**int front =-1, rear= -1**)

**Step 5** - Then implement main method by displaying menu of operations list and make suitable function calls to perform operation selected by the user on queue.

### **Enqueue (value)-Inserting value into the queue:**

In a queue data structure, enQueue() is a function used to insert a new element into the queue. In a queue, the new element is always inserted at **rear** position. The enQueue() function takes one integer value as parameter and inserts that value into the queue. We can use the following steps to insert an element into the queue...

**Step1-** Check whether **queue** is **FULL**. (**rear==SIZE-1**)

**Step2-** If it is **FULL**, then display "**Queue is FULL!!! Insertion is not possible!!!**" and terminate the function.

**Step3-** If it is **NOT FULL**, then increment **rear** value by one(**rear++**) and set **queue[rear]=value**.

### **Dequeue ()-Deleting a value from the Queue:**

In a queue data structure, deQueue() is a function used to delete an element from the queue. In a queue, the element is always deleted from **front** position. The deQueue() function does not take any value as parameter. We can use the following steps to delete an element from the queue...

**Step1-** Check whether **queue** is **EMPTY**. (**front==rear**)

**Step 2** - If it is **EMPTY**, then display "**Queue is EMPTY!!! Deletion is not possible!!!**" and terminate the function.

**Step 3** - If it is **NOT EMPTY**, then increment the **front** value by one (**front ++**).

Then display **queue[front]** as deleted element.

Then check whether both **front** and **rear** are equal(**front==rear**), if it **TRUE**, then set both **front** and **rear** to '**-1**'(**front=rear=-1**).

### **Display ()-Displays the elements of a Queue:**

We can use the following steps to display the elements of a queue...

**Step1**-Check whether **queue** is **EMPTY**. (**front==rear**)

**Step2**-If it is **EMPTY**, then display “**Queue is EMPTY!!!**” and terminate the function.

**Step3**-If it is **NOTEMPTY**, then define an integer variable '**i**' and set '**i=front+1**'.

**Step4**-Display '**queue[i]**' value and increment '**i**' value by one (**i++**).

Repeat the same until '**i**' value reaches to **rear** (**i <=rear**)

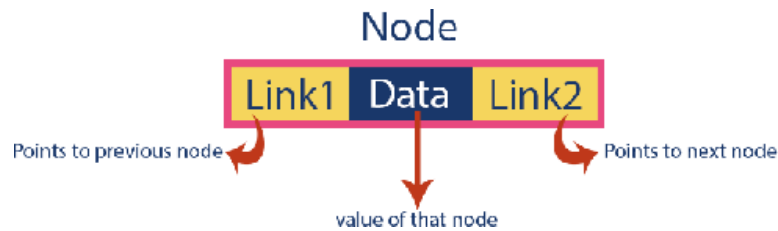
## Experiment

4. Write a program that uses functions to perform the following operations on doubly linked List.

i) Creation ii) Insertion iii) Deletion iv) Traversal

Double linked list is a sequence of elements in which every element has links to its previous element and next element in the sequence.

Every node in a double linked list contains three fields and they are shown in the following figure...



Here, 'link1' field is used to store the address of the previous node in the sequence, 'link2' field is used to store the address of the next node in the sequence and 'data' field is used to store the actual value of that node.

**Example:**



### Operations on Double Linked List:

In a double linked list, we perform the following operations...

1. Insertion
2. Deletion
3. Display

### Insertion:

In a double linked list, the insertion operation can be performed in three ways as follows...

1. Inserting At Beginning of the list
2. Inserting At End of the list
3. Inserting At Specific location in the list

### Inserting at Beginning of the list:

We can use the following steps to insert a new node at beginning of the double linked list...

**Step1**-Create a **newNode** with given value and **newNode**→**previous** as **NULL**.

**Step2**-Check whether list is **Empty**(**start**==**NULL**)

**Step3**- If it is **Empty** then, assign **NULL** to **new Node**→**next** and **newNode** to **start**.

**Step4**- If it is **not Empty** then, assign **start** to **newNode**→**next** and **newNode** to **start**

### Inserting At End of the list

We can use the following steps to insert a new node at end of the double linked list...

**Step1**-Create a **newNode** with given value and **newNode**→**next** as **NULL**.

**Step2**-Check whether list is **Empty** (**start**==**NULL**)

**Step3**-If it is **Empty**, then assign **NULL** to **new Node**→**previous** and **new Node** to **head**.

**Step4**- If it is **not Empty**, then, define a node pointer **temp** and initialize with **head**.

**Step5**-Keep moving the **temp** to its next node until it reaches to the last node in the list (until **temp** → **next** is equal to **NULL**).

**Step 6** - Assign **new Node** to **temp** → **next** and **temp** to **new Node**→**previous**

### Inserting At Specific location in the list (After a Node)

We can use the following steps to insert a new node after a node in the double linked list...

**Step1**-Create a **newNode** with given value.

**Step2**-Check whether list is **Empty**(**head**==**NULL**)

**Step 3** - If it is **Empty** then, assign **NULL** to both **newNode** → **previous** & **newNode** →**next** and set **newNode** to **head**.

**Step4**- If it is **not Empty** then, defines two node pointers **temp1**&**temp2** and initialize **temp1** with **head**.

**Step 5** - Keep moving the **temp1** to its next node until it reaches to the node after which we want to insert the **newNode** (until **temp1** → **data** is equal to **location**, here **location** is then node value after which we want to insert the **newNode**).

**Step 6** - Every time check whether **temp1** is reached to the last node. If it is reached to the last node then display '**Given node is not found in the list!!! Insertion not possible!!!**' and terminate the function. Otherwise move the **temp1** to next node.

**Step 7** – Assign temp1→next to temp2,newNodeto temp→ next,  
temp1tonewNode→previous,temp2 to newNode→ next  
newNodetotemp2→previous.

In a Double linked list, the deletion operation can be performed in three ways. They are as follows...

- 1) Deleting from Beginning of the list
- 2) Deleting from End of the list
- 3) Deleting a Specific Node

### **Deleting from Beginning of the list:**

We can use the following steps to delete an ode from beginning of the single linked list...

**Step1**-Check whet her list is **Empty** (**start==NULL**)

**Step2**-If it is **Empty** then, display '**List is Empty!!!Deletion is not possible**' and terminate the function.

**Step3**-Ifit is **NotEmpty** then, define a Node pointer '**temp**' and initialize with **start**.

**Step 4** - Check whether list is having only one node (**temp → next ==NULL**)

**Step 5** - If it is TRUE then set **start= NULL** and delete **temp** (Setting Empty list conditions)

**Step 6** - If it is FALSE then set **start= temp → next**, and delete **temp** (**free (temp)**).

### **Deleting from End of the list:**

We can use the following steps to delete a node from end of the single linked list...

**Step 1** - Check whether list is Empty (**start == NULL**)

**Step 2** - If it is Empty then, display '**List is Empty!!! Deletion is not possible**' and terminate the function.

**Step 3** - If it is Not Empty then, define two Node pointers '**temp**' and '**temp1**' and initialize '**temp**' with **start**.

**Step 4** - Check whether list has only one Node (**temp → next == NULL**)

**Step 5** - If it is TRUE. Then, set **start = NULL** and delete **temp**. And terminate the function. (Setting Empty list condition)

**Step 6** - If it is FALSE. Then, set '**temp1 = temp** ' and move **temp** to its next node. Repeat the same until it reaches to the last node in the list. (until **temp1 → next == NULL**)

**Step 7** - Finally, Set **temp1 → next = NULL** and delete **temp** (**free (temp)**).

### **Deleting a Specific Node from the list:**

**Step 1** - Create a newNode with given value

**Step 2** - Check whether list is Empty (start == NULL)

**Step 3** - If it is Empty then, set newNode → next = NULL and start = newNode.

**Step 4** - If it is Not Empty then, define a node pointer temp and initialize with start.

**Step 5** - Keep moving the temp to its next node until it reaches to the node after which we want to delete the Node

**Step 6**-Finally, Set p->next=temp->next temp->next->prev=p

**Step 7**-delete temp (free (temp)).

### **Traversing:**

- Assign the address of start pointer to a temp pointer.
- Display the information from the data field of each node.

**The function traverse () is used for traversing and displaying the information stored in the list from left to right.**

**5. Write a program that uses functions to perform the following operations on circular linked List.**

**i) Creation ii) Insertion iii) Deletion iv) Traversal**

Circular Singly linked list is similar to Single linked list, but the last node of the list contains a pointer to the first node of the list.

**Inserting At End of the list:**

We can use the following steps to insert a new node at end of the single linked list...

**Step1-**Create a **newNode** with given value and **newNode→next** as **NULL**.

**Step2-**Check whether list is **Empty** (**start==NULL**).

**Step3-**If it is **Empty** then, set **start=new Node**.

**Step4-** If it is **Not Empty** then defines an order pointer **temp** and initialize with **start**.

**Step5-**Keep moving the **temp** to its next node until it reaches to the last node in

The list (until **temp →next** is not equal to (**start**)).

**Step6-**Set **temp →next =new Node**.

**Step7-**Set **newNode→next =Start**

We can have circular singly linked list as well as circular doubly linked list.



## VII Searching & Sorting

### Demonstration

1. Write a C program that uses non recursive function to search for a Key value in a given list of integers using linear search method.

#### Algorithm:

Step 1: Start

Step 2: Read n, a[i], key values as integers

Step 3: Search the list

```
While (a[i] != key && i <= n)
```

```
 i = i + 1
```

```
 Repeat step 3
```

Step 4: Successful search

```
if(i = n + 1) then
```

```
 print "Element not found in the list"
```

```
 Return(0)
```

```
else
```

```
 print "Element found in the list"
```

```
 Return (i)
```

Step 6: Stop

#### Program:

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
void main()
```

```
{
```

```
 int i, a[20], n, key, flag = 0;
```

```
 clrscr();
```

```
 printf("Enter the size of an array \n");
```

```
 scanf("%d", &n);
```

```
 printf("Enter the array elements");
```

```
 for(i = 0; i < n; i++)
```

```
 {
```

```
 scanf("%d", &a[i]);
```

```
 }
```

```
 printf("Enter the key elements");
```

```
scanf("%d", &key);
for(i = 0; i < n; i++)
{
 if(a[i] == key)
 {
 flag = 1;
 break;
 }
}
if(flag == 1)
 printf("The key elements is found at location %d", i + 1);
else
 printf("The key element is not found in the array");
getch();
}
```

***Input & Output:***

Enter the size of an array 6

Enter the array elements 50 10 5 200 20 1

Enter the key element 1

The key Element is found at location 6

**2. Write a C program that uses non recursive function to search for a Key value in a given sorted list of integers using binary search method.**

***Algorithm:***

Step 1: Start

Step 2: Initialize

low = 1

high = n

Step 3: Perform Search

While(low <= high)

Step 4: Obtain index of midpoint of interval

Middle = (low + high) / 2

Step 5: Compare

if(X < K[middle])

high = middle - 1

else

print "Element found at position"

Return(middle)

goto: step 2

Step 6: Unsuccessful Search

print "Element found at position"

Return (middle)

Step 7: Stop

**Flowchart:**



**Program:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[20], i, n, key, low, high, mid;
clrscr();
```

```

printf("Enter the array elements in ascending order");
for(i = 0; i < n; i++)
{
scanf("%d", &a[i]);
}
printf("Enter the key element\n");
scanf("%d", &key);
low = 0;
high = n - 1;
while(high >= low)
{
mid = (low + high) / 2;
if(key == a[mid])
break;
else
{
if(key > a[mid])
low = mid + 1;
else
high = mid - 1;
}
}
if(key == a[mid])
printf("The key element is found at location %d", mid + 1);
else
printf("the key element is not found");
getch();
}

```

***Input & Output:***

Enter the size of the array 7

Enter the array elements in ascending order 23 45 68 90 100 789 890

Enter the key element 789

The key Element is found at location 6

**3. Write a C program that implements the Bubble sort method to sort a given list of integers in ascending order.**

***Algorithm:***

Step 1: Start

Step 2: Read n, a[i] values as integers

Step 3: for i: 1 to n do increment i by 1

begin

for j: 0 to n - i - 1 increment j by 1

begin

if(a[j] > a[j + 1])

begin

t = a[j];

a[j] = a[j + 1];

a[j + 1] = t;

end

end

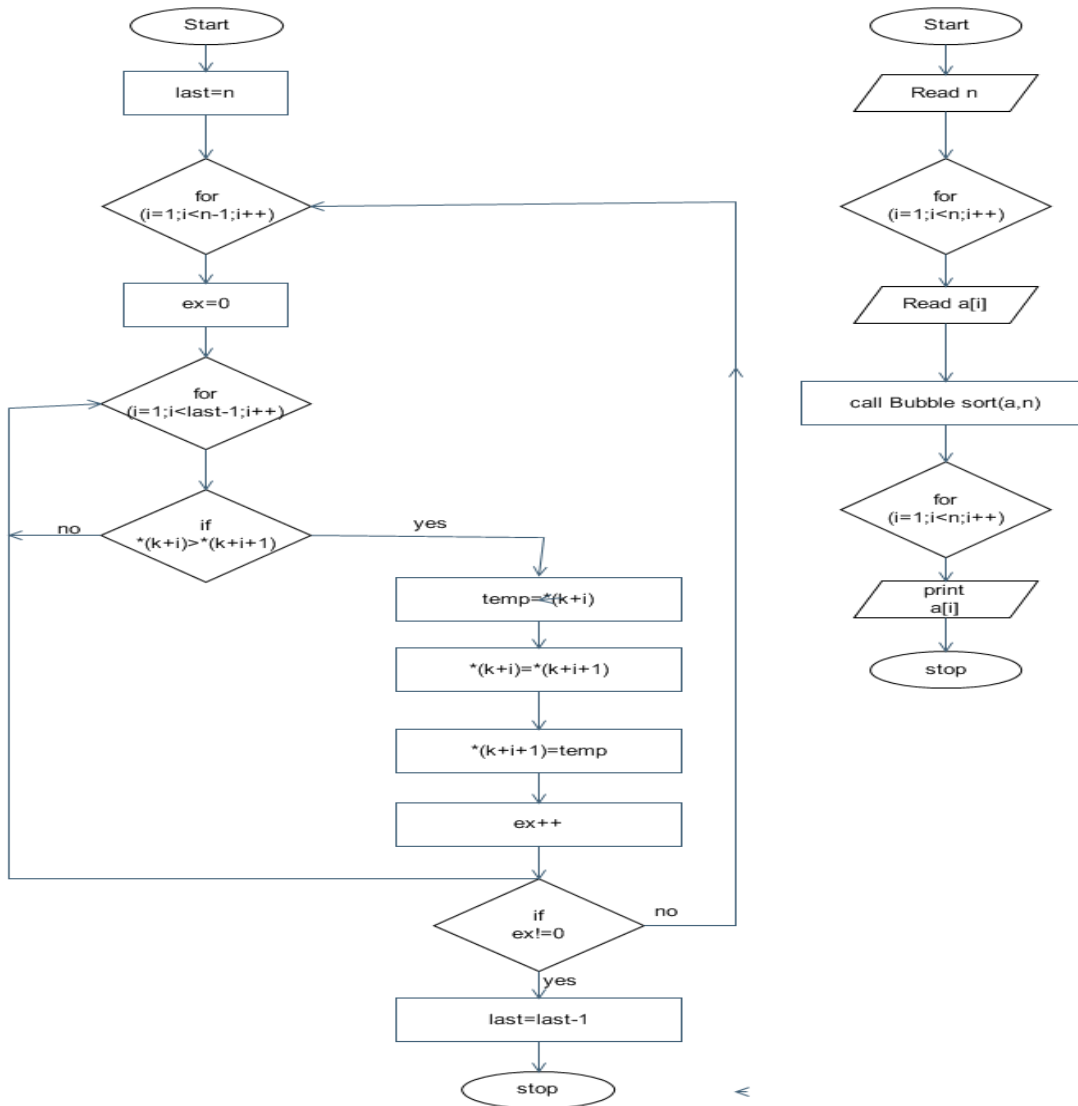
end

Step 4: for i: 0 to n

Print a[i]

Step 5: Stop

**Flowchart:**



**Experiment**

**5. Write a C program that sorts the given array of integers using selection sort in descending order**

**How selection sort works?**

*Lets consider the following array as an example: arr[] = {64, 25, 12, 22, 11}*

**First pass:**

- For the first position in the sorted array, the whole array is traversed from index 0 to 4 sequentially. The first position where **64** is stored presently, after traversing whole array it is clear that **11** is the lowest value.

|           |    |    |    |    |
|-----------|----|----|----|----|
| <b>64</b> | 25 | 12 | 22 | 11 |
|-----------|----|----|----|----|

- Thus, replace 64 with 11. After one iteration **11**, which happens to be the least value in the array, tends to appear in the first position of the sorted list.

|           |    |    |    |    |
|-----------|----|----|----|----|
| <b>11</b> | 25 | 12 | 22 | 64 |
|-----------|----|----|----|----|

**Second Pass:**

- For the second position, where 25 is present, again traverse the rest of the array in a sequential manner.

|    |           |    |    |    |
|----|-----------|----|----|----|
| 11 | <b>25</b> | 12 | 22 | 64 |
|----|-----------|----|----|----|

- After traversing, we found that **12** is the second lowest value in the array and it should appear at the second place in the array, thus swap these values.

|    |           |    |    |    |
|----|-----------|----|----|----|
| 11 | <b>12</b> | 25 | 22 | 64 |
|----|-----------|----|----|----|

**Third Pass:**

- Now, for third place, where **25** is present again traverse the rest of the array and find the third least value present in the array.

|    |    |           |    |    |
|----|----|-----------|----|----|
| 11 | 12 | <b>25</b> | 22 | 64 |
|----|----|-----------|----|----|

- While traversing, **22** came out to be the third least value and it should appear at the third place in the array, thus swap **22** with element present at third position.

|    |    |           |    |    |
|----|----|-----------|----|----|
| 11 | 12 | <b>22</b> | 25 | 64 |
|----|----|-----------|----|----|

**Fourth pass:**

- Similarly, for fourth position traverse the rest of the array and find the fourth least element in the array
- As **25** is the 4th lowest value hence, it will place at the fourth position.

|    |    |    |           |    |
|----|----|----|-----------|----|
| 11 | 12 | 22 | <b>25</b> | 64 |
|----|----|----|-----------|----|

**Fifth Pass:**

- At last the largest value present in the array automatically get placed at the last position in the array
- The resulted array is the sorted array.



|    |    |    |    |    |
|----|----|----|----|----|
| 11 | 12 | 22 | 25 | 64 |
|----|----|----|----|----|

Approach:

- Initialize minimum value(*min\_idx*) to location 0
- Traverse the array to find the minimum element in the array
- While traversing if any element smaller than **min\_idx** is found then swap both the values.
- Then, increment *min\_idx* to point to next element
- Repeat until array is sorted

**Problem Description:** Read an array of elements in random order, sort them using selection sort technique, then display array of elements in an order either ascending or descending. For example: Random elements of array are: 12 34 23 65 39, after selection sort the elements are: 12 23 34 39 65 or 65 39 34 23 12.

**Algorithm:**

**Step 1.** Select the first element of the list (i.e., Element at first position in the list).

**Step 2.** Compare the selected element with all other elements in the list.

**Step 3.** For every comparison, if any element is smaller than selected element (forAscending order), then these two are swapped.

**Step 4.** Repeat the same procedure with next position in the list till the entire list issorted.

**Sorting Logic:**

```

for(i=0; i<size; i++)
{
for(j=i+1; j<size; j++)
{
if(list[i] > list[j])
{
temp=list[i];
list[i]=list[j];
list[j]=temp;
}
}
}

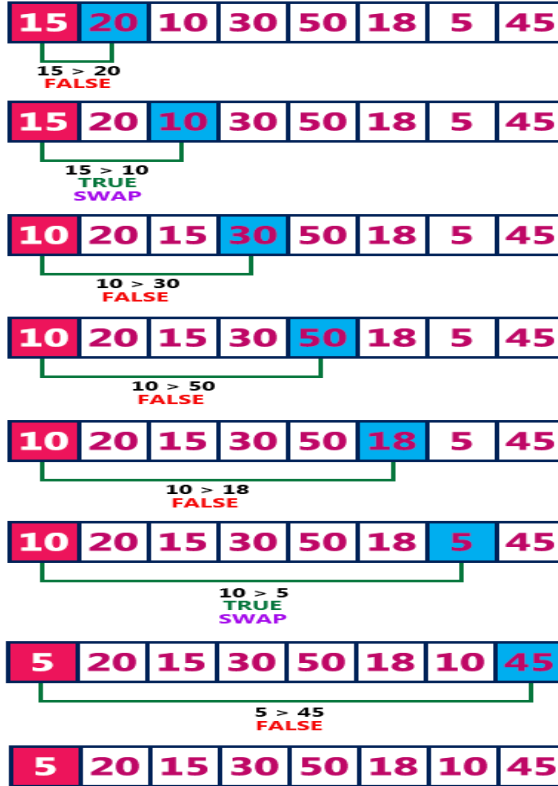
```

Consider the following unsorted list of elements...



**Iteration #1**

Select the first position element in the list, compare it with all other elements in the list and whenever we found a smaller element than the element at first position then swap those two elements.



**Iteration #2**

Select the second position element in the list, compare it with all other elements in the list and whenever we found a smaller element than the element at first position then swap those two elements.



**Iteration #3**

Select the third position element in the list, compare it with all other elements in the list and whenever we found a smaller element than the element at first position then swap those two elements.



**Iteration #4**

Select the fourth position element in the list, compare it with all other elements in the list and whenever we found a smaller element than the element at first position then swap those two elements.



**Iteration #5**

Select the fifth position element in the list, compare it with all other elements in the list and whenever we found a smaller element than the element at first position then swap those two elements.



**Iteration #6**

Select the sixth position element in the list, compare it with all other elements in the list and whenever we found a smaller element than the element at first position then swap those two elements.

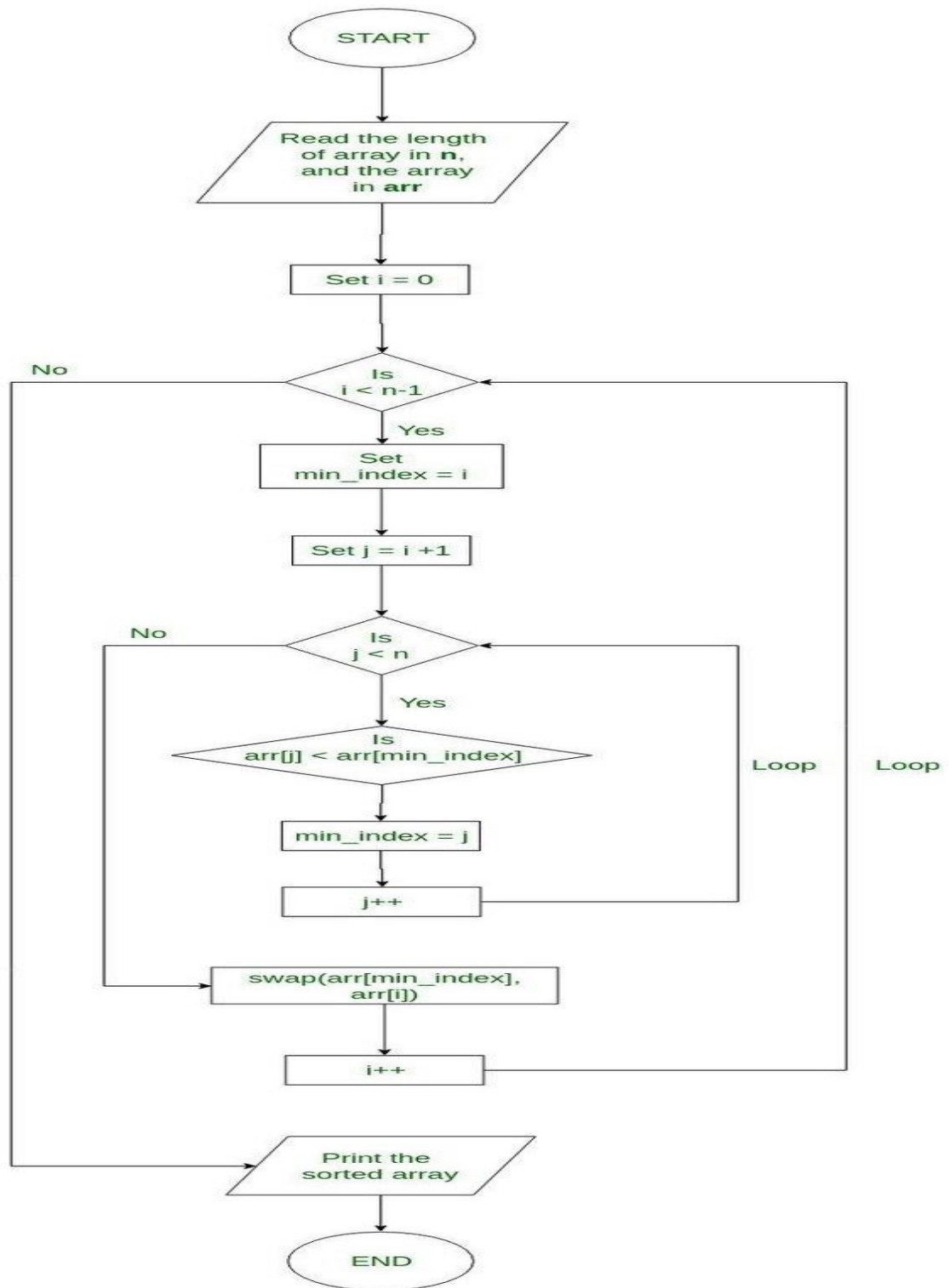


**Iteration #7**

Select the seventh position element in the list, compare it with all other elements in the list and whenever we found a smaller element than the element at first position then swap those two elements.



←  
**Final sorted list**



**Flowchart for Selection Sort**

## 6. Write a C program that sorts the given array of integers using insertion sort in ascending order

### Characteristics of Insertion Sort:

- This algorithm is one of the simplest algorithm with simple implementation
- Basically, Insertion sort is efficient for small data values
- Insertion sort is adaptive in nature, i.e. it is appropriate for data sets which are already partially sorted.

### Working of Insertion Sort algorithm:

Consider an example:  $arr[]: \{12, 11, 13, 5, 6\}$

|    |    |    |   |   |
|----|----|----|---|---|
| 12 | 11 | 13 | 5 | 6 |
|----|----|----|---|---|

#### **First Pass:**

|    |    |    |   |   |
|----|----|----|---|---|
| 12 | 11 | 13 | 5 | 6 |
|----|----|----|---|---|

- Initially, the first two elements of the array are compared in insertion sort.
- Here, 12 is greater than 11 hence they are not in the ascending order and 12 is not at its correct position. Thus, swap 11 and 12.
- So, for now 11 is stored in a sorted sub-array.

|    |    |    |   |   |
|----|----|----|---|---|
| 11 | 12 | 13 | 5 | 6 |
|----|----|----|---|---|

#### **Second Pass:**

- Now, move to the next two elements and compare them

|    |    |    |   |   |
|----|----|----|---|---|
| 11 | 12 | 13 | 5 | 6 |
|----|----|----|---|---|

- Here, 13 is greater than 12, thus both elements seems to be in ascending order, hence, no swapping will occur. 12 also stored in a sorted sub-array along with 11

#### **Third Pass:**

- Now, two elements are present in the sorted sub-array which are 11 and 12
- Moving forward to the next two elements which are 13 and 5

|    |    |    |   |   |
|----|----|----|---|---|
| 11 | 12 | 13 | 5 | 6 |
|----|----|----|---|---|

- Both 5 and 13 are not present at their correct place so swap them

|    |    |   |    |   |
|----|----|---|----|---|
| 11 | 12 | 5 | 13 | 6 |
|----|----|---|----|---|

- After swapping, elements 12 and 5 are not sorted, thus swap again

|    |   |    |    |   |
|----|---|----|----|---|
| 11 | 5 | 12 | 13 | 6 |
|----|---|----|----|---|

- Here, again 11 and 5 are not sorted, hence swap again

|   |    |    |    |   |
|---|----|----|----|---|
| 5 | 11 | 12 | 13 | 6 |
|---|----|----|----|---|

- here, it is at its correct position

**Fourth Pass:**

- Now, the elements which are present in the sorted sub-array are 5, 11 and 12

- Moving to the next two elements 13 and 6

|   |    |    |    |   |
|---|----|----|----|---|
| 5 | 11 | 12 | 13 | 6 |
|---|----|----|----|---|

- Clearly, they are not sorted, thus perform swap between both

|   |    |    |   |    |
|---|----|----|---|----|
| 5 | 11 | 12 | 6 | 13 |
|---|----|----|---|----|

- Now, 6 is smaller than 12, hence, swap again

|   |    |   |    |    |
|---|----|---|----|----|
| 5 | 11 | 6 | 12 | 13 |
|---|----|---|----|----|

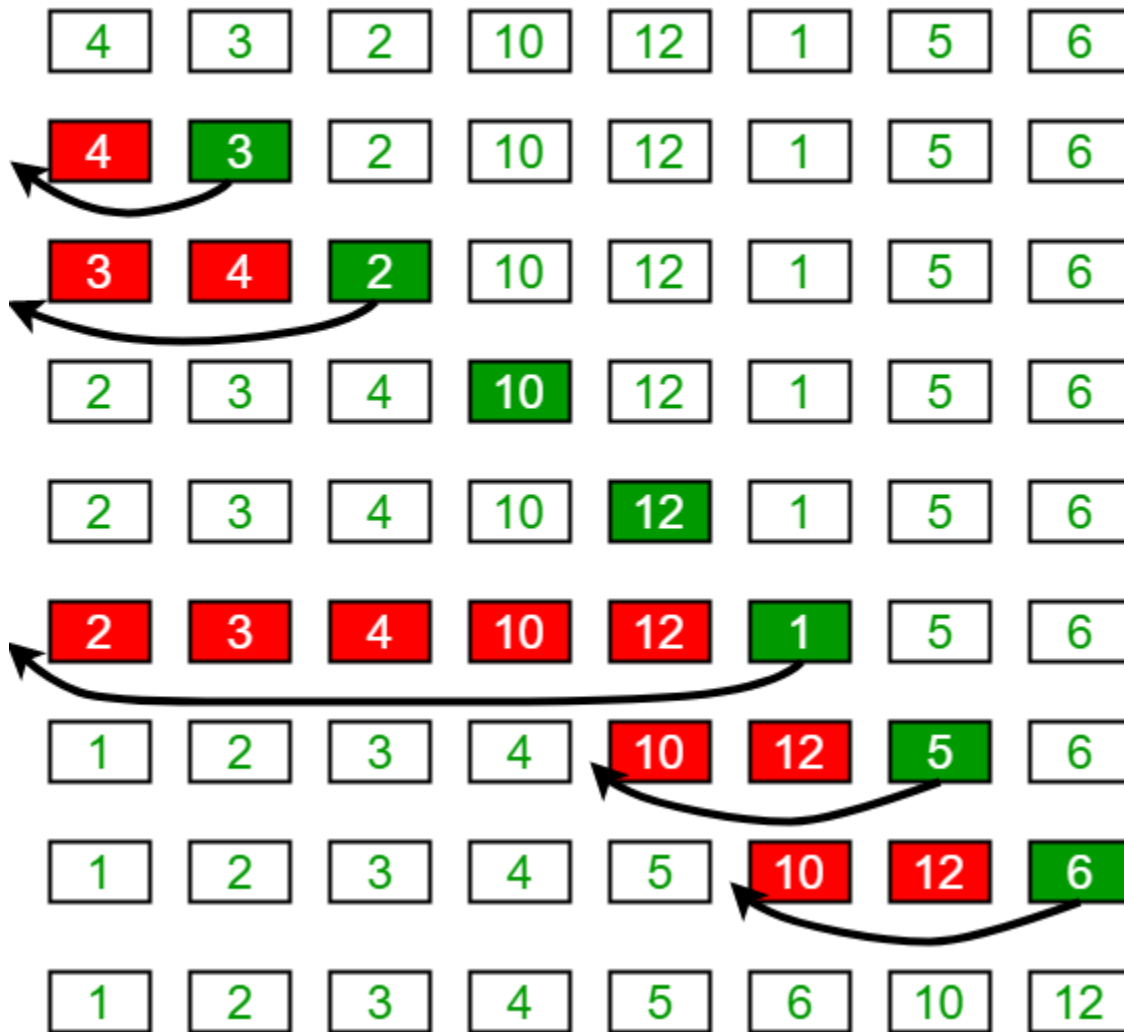
- Here, also swapping makes 11 and 6 unsorted hence, swap again

|   |   |    |    |    |
|---|---|----|----|----|
| 5 | 6 | 11 | 12 | 13 |
|---|---|----|----|----|

- Finally, the array is completely sorted.

**Illustrations:**

## Insertion Sort Execution Example



### Insertion Sort Algorithm

To sort an array of size N in ascending order:

- Iterate from  $arr[1]$  to  $arr[N]$  over the array.
- Compare the current element (key) to its predecessor.
- If the key element is smaller than its predecessor, compare it to the elements before.
- Move the greater elements one position up to make space for the swapped element.

**Problem Description:** Insertion Sort is basically insertion of an element from a random set of numbers, to its correct position where it should actually be, by shifting the other elements if required.

**Algorithm:**

**Step 1.** We will store the random set of numbers in an array.

**Step 2.** We will traverse this array and insert each element of this array, to its correct position where it should actually be, by shifting the other elements on the left if required. **Step 3.** The first element in the array is considered as sorted, even if it is an unsorted array. The array is sub-divided into two parts, the first part holds the first element of the array which is considered to be sorted and second part contains all the remaining elements of array.

**Step 4.** With each iteration one element from the second part is picked and inserted into the first part of array at its correct position by shifting the existing elements if required. **Step 5.** This goes until the last element in second part of array is placed in correct position in the output array.

**Step 6.** Now, we have the array in sorted order.

**Insertion Sort Logic**

```
//Insertion sort logic
for
i = 1 to size-1 { temp =
list[i];
 j = i-1;
while ((temp < list[j]) && (j > 0)) {list[j]
= list[j-1];
 j = j - 1;
}
list[j] = temp; }
```

### Example:

Consider the following unsorted list of elements...

|    |    |    |    |    |    |   |    |
|----|----|----|----|----|----|---|----|
| 15 | 20 | 10 | 30 | 50 | 18 | 5 | 45 |
|----|----|----|----|----|----|---|----|

Assume that sorted portion of the list is empty and all elements in the list are in unsorted portion of the list as shown in the figure below...

| Sorted | Unsorted               |
|--------|------------------------|
|        | 15 20 10 30 50 18 5 45 |

Move the first element 15 from unsorted portion to sorted portion of the list.

| Sorted | Unsorted            |
|--------|---------------------|
| 15     | 20 10 30 50 18 5 45 |

To move element 20 from unsorted to sorted portion, Compare 20 with 15 and insert it at correct position

| Sorted | Unsorted         |
|--------|------------------|
| 15 20  | 10 30 50 18 5 45 |

To move element 10 from unsorted to sorted portion, Compare 10 with 20 and it is smaller so swap. Then compare 10 with 15 again smaller swap. And 10 is inserted at its correct position in sorted portion of the list.

| Sorted   | Unsorted      |
|----------|---------------|
| 10 15 20 | 30 50 18 5 45 |

To move element 30 from unsorted to sorted portion, Compare 30 with 20, 15 and 10. And it is larger than all these so 30 is directly inserted at last position in sorted portion of the list.

| Sorted      | Unsorted   |
|-------------|------------|
| 10 15 20 30 | 50 18 5 45 |

To move element 50 from unsorted to sorted portion, Compare 50 with 30, 20, 15 and 10. And it is larger than all these so 50 is directly inserted at last position in sorted portion of the list.

| Sorted         | Unsorted |
|----------------|----------|
| 10 15 20 30 50 | 18 5 45  |

To move element 18 from unsorted to sorted portion, Compare 18 with 30, 20 and 15. Since 18 is larger than 15, move 20, 30 and 50 one position to the right in the list and insert 18 after 15 in the sorted portion.

| Sorted            | Unsorted |
|-------------------|----------|
| 10 15 18 20 30 50 | 5 45     |

To move element 5 from unsorted to sorted portion, Compare 5 with 50, 30, 20, 18, 15 and 10. Since 5 is smaller than all these elements, move 10, 15, 18, 20, 30 and 50 one position to the right in the list and insert 5 at first position in the sorted list.

| Sorted              | Unsorted |
|---------------------|----------|
| 5 10 15 18 20 30 50 | 45       |

To move element 45 from unsorted to sorted portion, Compare 45 with 50 and 30. Since 45 is larger than 30, move 50 one position to the right in the list and insert 45 after 30 in the sorted list.

| Sorted                 | Unsorted |
|------------------------|----------|
| 5 10 15 18 20 30 45 50 |          |

Unsorted portion of the list has become empty. So we stop the process. And the final sorted list of elements is as follows...

|   |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|
| 5 | 10 | 15 | 18 | 20 | 30 | 45 | 50 |
|---|----|----|----|----|----|----|----|



