



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



PROGRAMMING FOR PROBLEM SOLVING LAB

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

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 computerscience education and to prepare students for professional career and/orhigher studies globally.
- To develop self learning abilities and professional ethics to serve the society.

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 : PROGRAMMING FOR PROBLEM SOLVING LAB

Subject Code : A405502

Class & Branch/ Specialization : I B.Tech. I – Semester

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

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 : PROGRAMMING FOR PROBLEM SOLVING LAB

Subject Code : A405502

**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	To formulate the algorithms for simple problems and translate given algorithms to a working and correct program.
2	To correct syntax errors as reported by the compilers identify and correct logical errors encountered during execution
3	To represent and manipulate data with arrays, strings and structures and use pointers of different types
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> <ol style="list-style-type: none"> 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).
WEEK - 2	<p>Experiment</p> <ol style="list-style-type: none"> 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. Write a C program that prints a multiplication table for a given number and the number of rows in the table. (For example, for a number 5 and rows = 3, the output should be: 5 x 1 = 5, 5 x 2 = 10, 5 x 3 = 15....) 9. Write a program that shows the binary equivalent of a given positive number between 0 to 255. 10. Write a program that asks the user to enter the total time elapsed,

	in seconds, since an event and converts the time to hours, minutes and seconds. The time should be displayed as hours:minutes:seconds. [Hint: Use the remainder operator]
WEEK - 3	<p>II. Expression Evaluation Demonstration</p> <ol style="list-style-type: none"> 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/sec² ($= 9.8$ m/s²)). 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. 3. Write a C program, which takes two integer operands and one operator from the user, performs the operation and then prints the result. (Consider the operators +, -, *, /, % and use Switch Statement). 4. Write a program that finds if a given number is a prime number 5. Write a C program to find the sum of individual digits of a positive integer and test given number is palindrome. 6. 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. 7. 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.
WEEK - 4	<p>Experiment</p> <ol style="list-style-type: none"> 8. Write a C program to generate all the prime numbers between 1 and n, where n is a value supplied by the user. 9. Write a C program to find the roots of a Quadratic equation.
WEEK - 5	<p>III. Iterative statements Demonstration</p> <ol style="list-style-type: none"> 1. Input an integer (5 digits or fewer) containing only 0s and 1s (i.e., a “binary” integer) and print its decimal equivalent. [Hint: Use the remainder and division operators to pick off the “binary” number’s

digits one at a time from right to left. Just as in the decimal number system, in which the rightmost digit has a positional value of 1, and the next digit left has a positional value of 10, then 100, then 1000, and so on, in the binary number system the rightmost digit has a positional value of 1, the next digit left has a positional value of 2, then 4, then 8, and so on. Thus the decimal number 234 can be interpreted as $4 * 1 + 3 * 10 + 2 * 100$. The decimal equivalent of binary 1101 is $1 * 1 + 0 * 2 + 1 * 4 + 1 * 8$ or $1 + 0 + 4 + 8$ or 13.]

2. Armstrong numbers are numbers that are equal to the sum of their digits raised to power of the number of digits in them. The number 153, for example, equals $1^3 + 5^3 + 3^3$. Thus it is an Armstrong number. Write a program to display all three-digit Armstrong numbers.

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

4. 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?

5. Write a C program to calculate the following, where x is a fractional value. $1 - x/2 + x^2/4 - x^3/6 \dots\dots$

6. Write a C program to read in two numbers, x and n, and then compute the sum of this geometric progression:

$1 + x + x^2 + x^3 + \dots\dots\dots + x^n$. For example: if n is 3 and x is 5, then the program computes $1 + 5 + 25 + 125$.

7. 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
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

```

Experiment

8. Write a program that reads three nonzero integer values and determines and prints whether they could represent the sides of a triangle.

9. Write a program that reads three nonzero integers and determines and prints whether they could be the sides of a right triangle

10. Write a program that reads a nonnegative integer and computes and prints its factorial

WEEK - 6

	<p>11. Write a program that estimates the value of the mathematical constant e by using the formula:</p> $e^1 = 1 + \frac{1}{1!} + \frac{1^2}{2!} + \frac{1^3}{3!} + \dots$ <p>12. Write a program that computes the value of e^x by using the formula</p> $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots, -\infty < x < \infty$
WEEK -7	<p>IV. Arrays, Pointers, and Functions Demonstration</p> <p>1. Write a C program to find the minimum, maximum and average in an array of integers.</p> <p>2. Write a function to compute mean, variance, Standard Deviation, sorting of n elements in a single dimension array.</p> <p>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 with memory dynamically allocated for the new matrix as row and column counts may not be the same.</p> <p>4. Write C programs that use both recursive and non-recursive functions</p> <p>5. To find the factorial of a given integer.</p>
WEEK - 8	<p>Experiment</p> <p>6. Write a C program to find the GCD (greatest common divisor) of two given integers.</p> <p>7. Write a C program to compute x^n</p> <p>8. Write a program for reading elements using a pointer into an array and display the values using the array.</p> <p>9. Write a program for display values reverse order from an array using a pointer.</p> <p>10. Write a program through a pointer variable to sum of n elements from an array.</p>
WEEK - 9	<p>V. Files Demonstration</p> <p>1. Write a C program to display the contents of a file to standard output device.</p> <p>2. Write a C program which copies one file to another, replacing all lowercase characters with their uppercase</p>

	<p>equivalents.</p> <p>3. Write a C program to count the number of times a character occurs in a text file. The file name and the character are supplied as command line arguments.</p>
WEEK - 10	<p>Experiment</p> <p>4. Write a C program that does the following: It should first create a binary file and store 10 integers, where the file name and 10 values are given in the command line. (hint: convert the strings using atoi function) Now the program asks for an index and a value from the user and the value at that index should be changed to the new value in the file. (hint: use fseek function) The program should then read all 10 values and print them back</p> <p>5. Write a C program to merge two files into a third file (i.e., the contents of the first file followed by those of the second are put in the third file).</p>
WEEK - 11	<p>VI. Strings Demonstration</p> <p>1. Write a C program to convert a Roman numeral ranging from I to L to its decimal equivalent.</p> <p>2. Write a C program that converts a number ranging from 1 to 50 to Roman equivalent c.</p> <p>3. Write a C program that uses functions to perform the following operations:</p> <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 - 12	<p>Experiment</p> <p>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.)</p> <p>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.</p> <p>6. Write a C program to count the lines, words and characters in a given text.</p>
WEEK-13	<p>VII. Sorting and Searching: Demonstration</p> <p>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.</p> <p>2. Write a C program that uses non recursive function to search for a</p>

	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	Experiment 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 6. Write a C program that sorts a given array of names
PROJECTS	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 C 7th 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	Page No.
1.	Write a C program to compute simple and compound interest .	
2.	Write a C program that declares Class awarded for a given percentage of marks, where mark = 70% = Distinction. (Read percentage from standard input.)	
3.	Write a C program that prints a multiplication table for a given number and the number of rows in the table. (For example, for a number 5 and rows = 3, the output should be: 5 x 1 = 5, 5 x 2 = 10, 5 x 3 = 15....)	
4.	Write a C program that shows the binary equivalent of a given positive number between 0 to 255	
5.	Write a C program that asks the user to enter the total time elapsed, in seconds, since an event and converts the time to hours, minutes and seconds. The time should be displayed as hours:minutes:seconds. [Hint: Use the remainder operator]	
6.	Write a C program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.	
7.	Write a C program to find the roots of a Quadratic equation	
8.	Write a C program that reads three nonzero integer values and determines and prints whether they could represent the sides of a triangle.	
9.	Write a C program that reads three nonzero integers and determines and prints whether they could be the sides of a right triangle	
10.	Write a C program that reads a nonnegative integer and computes and prints its factorial	
11.	Write a C program that estimates the value of the mathematical constant e by using the formula: $e^1 = 1 + \frac{1}{1!} + \frac{1^2}{2!} + \frac{1^3}{3!} + \dots$	
12.	Write a C program that computes the value of ex by using the formula $e^x = 1 + x/1! + x^2/2! + x^3/3! + \dots, -\infty < x < \infty$	

13.	Write a C program to find the GCD (greatest common divisor) of two given integers.	
14.	Write a C program to compute x^n	
15.	Write a C program for reading elements using a pointer into an array and display the values using the array.	
16.	Write a C program for display values reverse order from an array using a pointer	
17.	Write a C program through a pointer variable to sum of n elements from an array.	
18.	Write a C program that does the following: It should first create a binary file and store 10 integers, where the file name and 10 values are given in the command line. (hint: convert the strings using atoi function)Now the program asks for an index and a value from the user andthe value at that index should be changed to the new value in the file. (hint: use fseek function)The program should then read all 10 values and print them back	
19.	Write a C program to merge two files into a third file (i.e., the contents of the first file followed by those of the second are put in the third file).	
20.	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.)	
21.	Write a C program that displays the position of a character ch in the string S or – 1 if S doesn't contain ch.	
22.	Write a C program to count the lines, words and characters in a given text.	
23.	Write a C program that sorts the given array of integers using selection sort in descending order	
24.	Write a C program that sorts the given array of integers using insertion sort in ascending order	
25.	Write a C program that sorts a given array of names	

Week1 Demonstration

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

24-hour time 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>

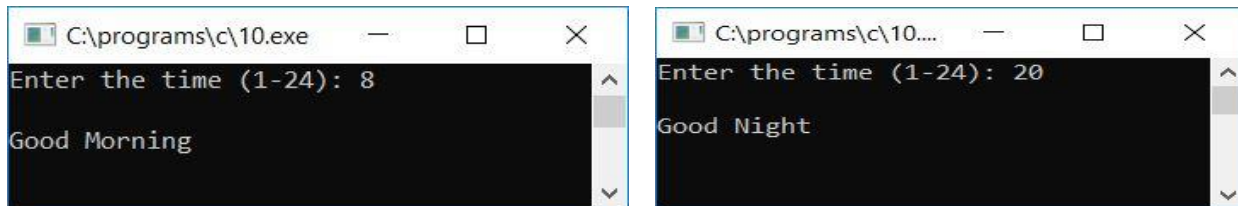
int main()
{
int t;
printf("Enter the time (1-24): ");
scanf("%d", &t);
printf("\n");
if(t>0 && t<=3)
printf("Good Night");
else if(t>3 && t<12)
printf("Good Morning");
else if(t==12)
printf("Good Noon");
else if(t>12 && t<=15)
printf("Good AfterNoon");
```

```

else if(t>15 && t<20)
printf("Good Evening");
else if(t>=20 && t<=24)
printf("Good Night");
else
printf("Unknown 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

```

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

```


4. 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 ‘a’ 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:

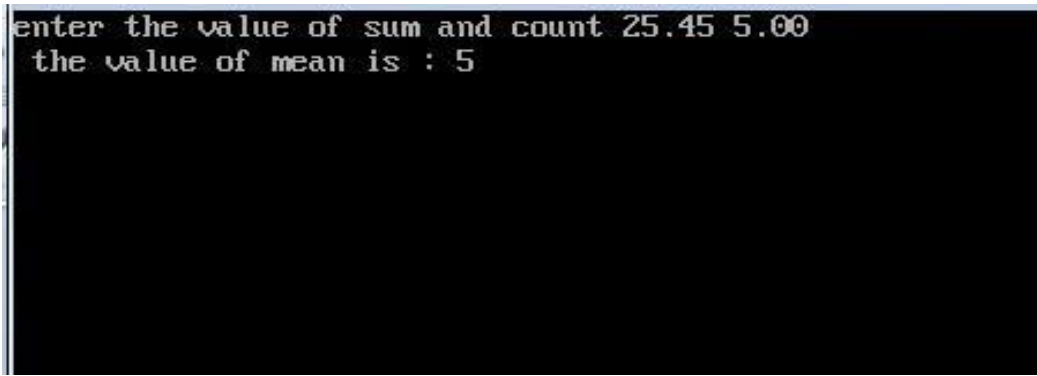
```
float f = 6.78;  
  
int k = (int) f; // Manual casting: float to int
```

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
```

5. 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.

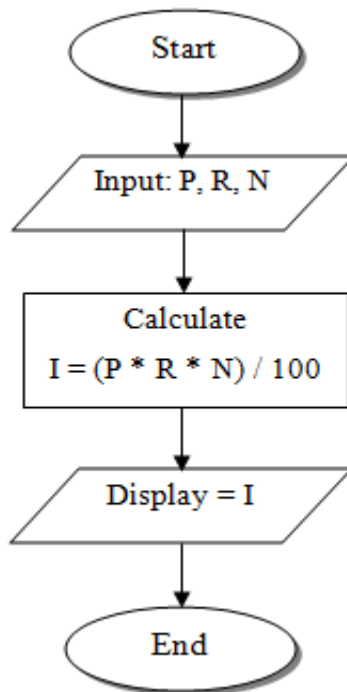
**Step 3:** Calculate Interest using formula  $SI = ((\text{amount} * \text{rate} * \text{time}) / 100)$ ,  $CI = P * (1 + r/100)^n - p$

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

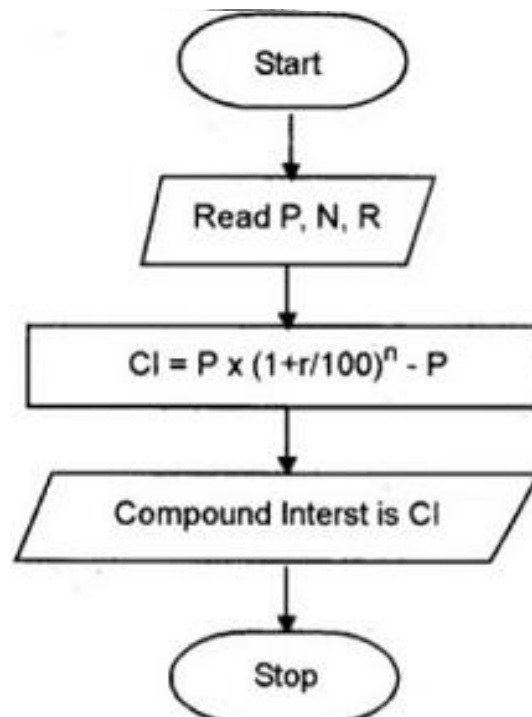
**Step 5:** Stop

#### 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:**

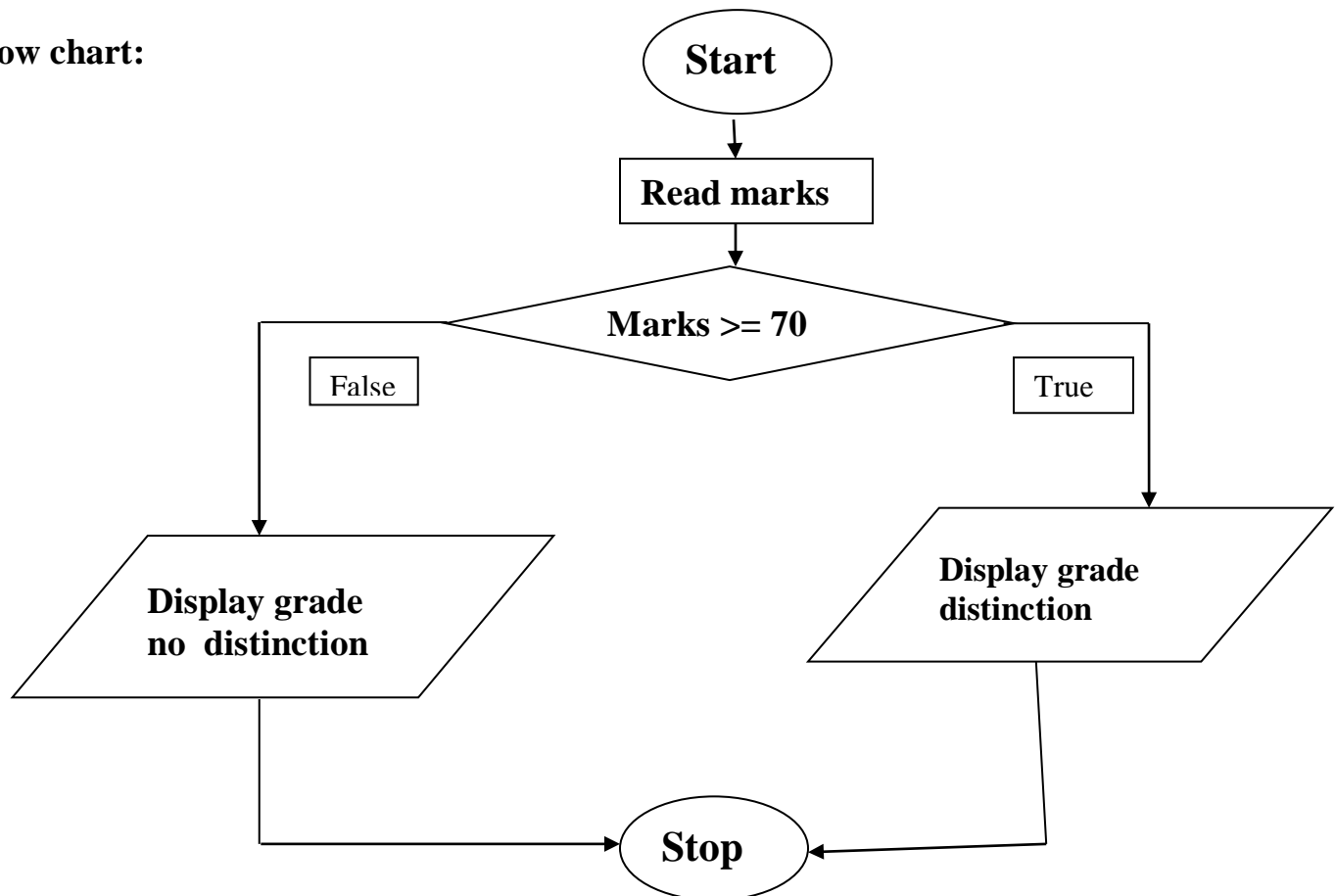
**Step 1 :** start

**Step 2 :** read marks or Percentage

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

**Step 4 :** stop.

**Flow chart:**



### Program 8:

**8. Write a C program that prints a multiplication table for a given number and the number of rows in the table.**

**(For example, for a number 5 and rows = 3, the output should be:  
 $5 \times 1 = 5$ ,  $5 \times 2 = 10$ ,  $5 \times 3 = 15$ ....)**

**Problem Description:** C program that prints a multiplication table for a given number and the number of rows in the table.

#### Algorithm:

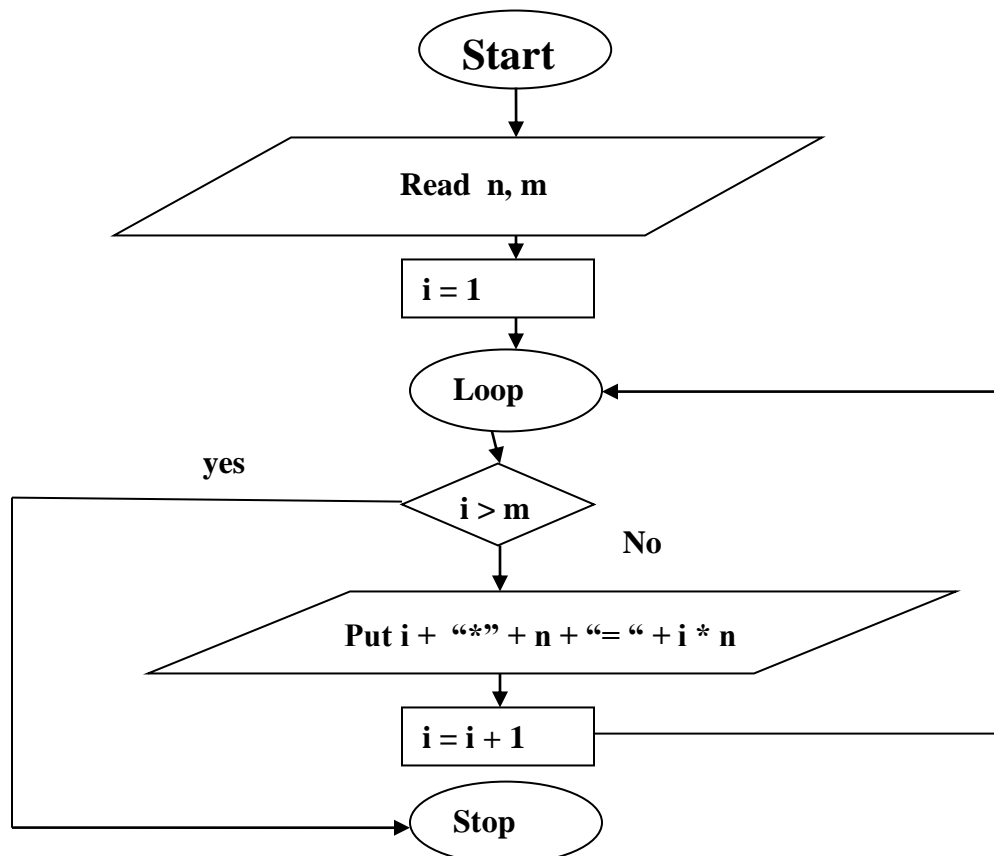
**Step 1:** start

**Step 2 :** Get input n and m

**Step 3:** print the multiplication table of n till m

**Step 4 :** stop

#### Flowchart:





**Program 9:**

**9. Write a C program that shows the binary equivalent of a given positive number between 0 to 255.**

**Problem Description:** The binary equivalent of a given positive number between 0 to 255

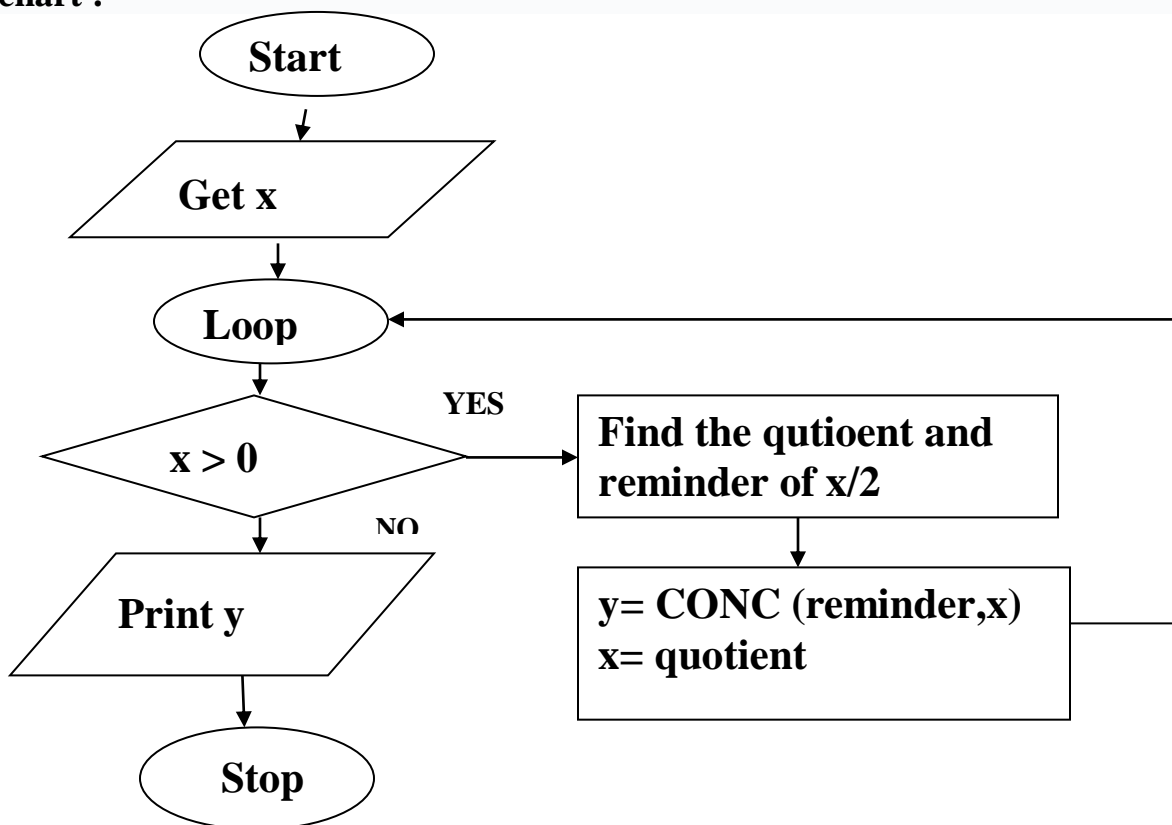
**Algorithm:**

**Step 1:** Divide the number by 2 and find the remainder, then store the remainder in an array.

**Step 2:** Divide the number by 2.

**Step 3:** Repeat the above two steps until the number is greater than zero.

**Flow chart :**



**Program 10:**

**10. Write a program that asks the user to enter the total time elapsed, in seconds, since an event and converts the time to hours, minutes and seconds. The time should be displayed as hours:minutes:seconds. [Hint: Use the remainder operator]**

**Problem Description:** user to enter the total time elapsed, in seconds, since an event and converts the time to hours, minutes and seconds

**Algorithm:**

**Step 1:** Start

**Step 2:** Take the no. of hours as an input into an integer variable that denotes hours.

**Step 3:** Take the no. of minutes as an input into an integer variable that denotes minutes.

**Step 4:** Take the no. of seconds as an input into an integer variable that denotes seconds.

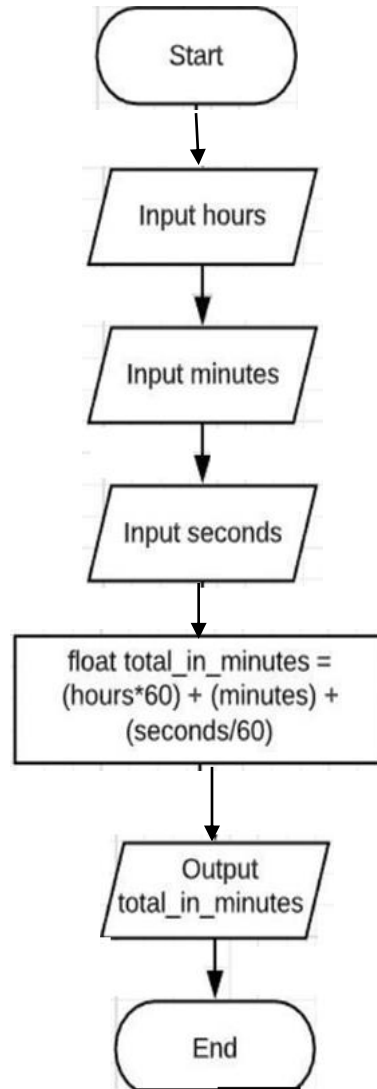
**Step 5:** As 1 hr = 60 min, and 1 second = 1/60 minutes, convert all of them into minutes and store the result into a floating variable that denotes the total in minutes.

$\text{total\_in\_minutes} = (\text{hours} * 60) + \text{minutes} + (\text{seconds} / 60)$

**Step 6:** Print the variable that denotes the total in minutes.

**Step 7:** End.

**Flowchart:**



## VIVA-QUESTIONS:

1. Which is valid C expression?

- A. `int my_num = 100,000;`
- B. `int my_num = 100000;`
- C. `int my num = 1000;`
- D. `int $my_num = 10000;`

2. Which of the following is the correct order of evaluation for the below expression?

$$z = x + y * z / 4 \% 2 - 1$$

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

3. The precedence of arithmetic operators is (from highest to lowest)?

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

4. Which of the following is not an arithmetic operation?

- A. `a *= 20;`
- B. `a /= 30;`
- C. `a %= 40;`
- D. `a != 50;`

5. Which of the following data type will throw an error on modulus operation(%)?

- A. `char`
- B. `short`

- C. float
- D. int

6. What is the output of the following code?

```
int main()
{
 float f1 = 0.1;
 if (f1 == 0.1)
 printf("equal\n");
 else
 printf("not equal\n");
}
```

- A. equal
- B. not equal
- C. Output depends on compiler
- D. None of the mentioned

### Answer & Explanation

---

Answer: Option B

Explanation:

By default is of type double which has different representation than float resulting in inequality even after conversion.

7. What is the output of this C code?

```
int main()
{
 int var = 010;
 printf("%d", var);
}
```

- A. 2
- B. 8
- C. 9
- D. 10

8. What is the output of this C code?

```
void main()
{
 int x = 4.3 % 2;
 printf("Value of x is %d", x);
}
```

- A. Value of x is 1.3
- B. Value of x is 2
- C. Value of x is 0.3
- D. Compile time error

## 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 value
double 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,smmonth,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[smonth]) ;
//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

**3. Write a C program, which takes two integer operands and one operator from the user, performs the operation and then prints the result. (Consider the operators +, -, \*, /, % and use Switch Statement).**

**Program:**

```
#include <stdio.h>
#include <conio.h>
void main()
{
int a, b, c;
char ch;
clrscr() ;
 printf("Enter your operator(+, -, /, *, %)\n");
 scanf("%c", &ch);
 printf("Enter the values of a and b\n");
 scanf("%d%d", &a, &b);
 switch(ch)
 {
case '+': c = a + b;
printf("addition of two numbers is %d", c);
break;
case '-': c = a - b;
printf("substraction of two numbers is %d", c);
break;
case '*': c = a * b;
printf("multiplication of two numbers is %d", c);
break;
case '/': c = a / b;
```

```
printf("remainder of two numbers is %d", c);
break;
case '%': c = a % b;
printf("quotient of two numbers is %d", c);
break;
default: printf("Invalid operator");
break;
}
getch();
}
```

Output:

```
Enter your operator(+, -, /, *, %)
+
Enter the values of a and b
4 5
addition of two numbers is 9
```

#### 4. Write a program that finds if a given number is a prime number

```
#include<stdio.h>

int main()
{
 int n,i,flag=0;
 printf("\nEnter a number:");
 scanf("%d",&n);
 for(i=2;i<=n/2;i++)
 {
 if(n%i==0)
 {
 flag=1;
 break;
 }
 }
 If(flag==0)
 printf("%d is a prime number",n);
 else
 printf("%d is not a prime number",n);
 return(0); }
```

#### output

Enter a number:29

29 is a prime number

**5. Write a C program to find the sum of individual digits of a positive integer and test given number is palindrome.**

```
#include<stdio.h>

main()
{
int number=0, digit=0;
sum=0;
clrscr();
printf("Enter any number\n");
scanf("%d",&number);
while(number!=0)
{
digit = number % 10;
sum = sum + digit;
number=number/10;
}
printf("Sum of individual digits of a given number is %d", sum);
getch();
}
```

**Output:**

```
Enter any number
105
Sum of individual digits of a given number is 6_
```

### Test given number is palindrome or not.

Program:

```
#include <stdio.h>

void main()
{
 int number, t, rev=0, rmdr;

 printf("Please enter a number to check Palindrome : "); scanf("%d",&number);
 printf("\nEntered number: %d", number); t = number;
 while (number > 0)
 {
 rmdr = number%10; rev = rev*10 + rmdr; number = number/10;
 }
 printf("\nReversed number: %d", rev); if(t == rev)
 {
 printf("\nEntered number %d is a palindrome", t);
 }
 else
 {
 printf("\nEntered number %d is not a palindrome", t);
 }
}
```

### Output:

```
Please enter a number to check Palindrome : 141
Entered number: 141
Reversed number: 141
Entered number 141 is a palindrome_
```



**6. 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

**7. 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 a
choice 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: -1.000000

2

## 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

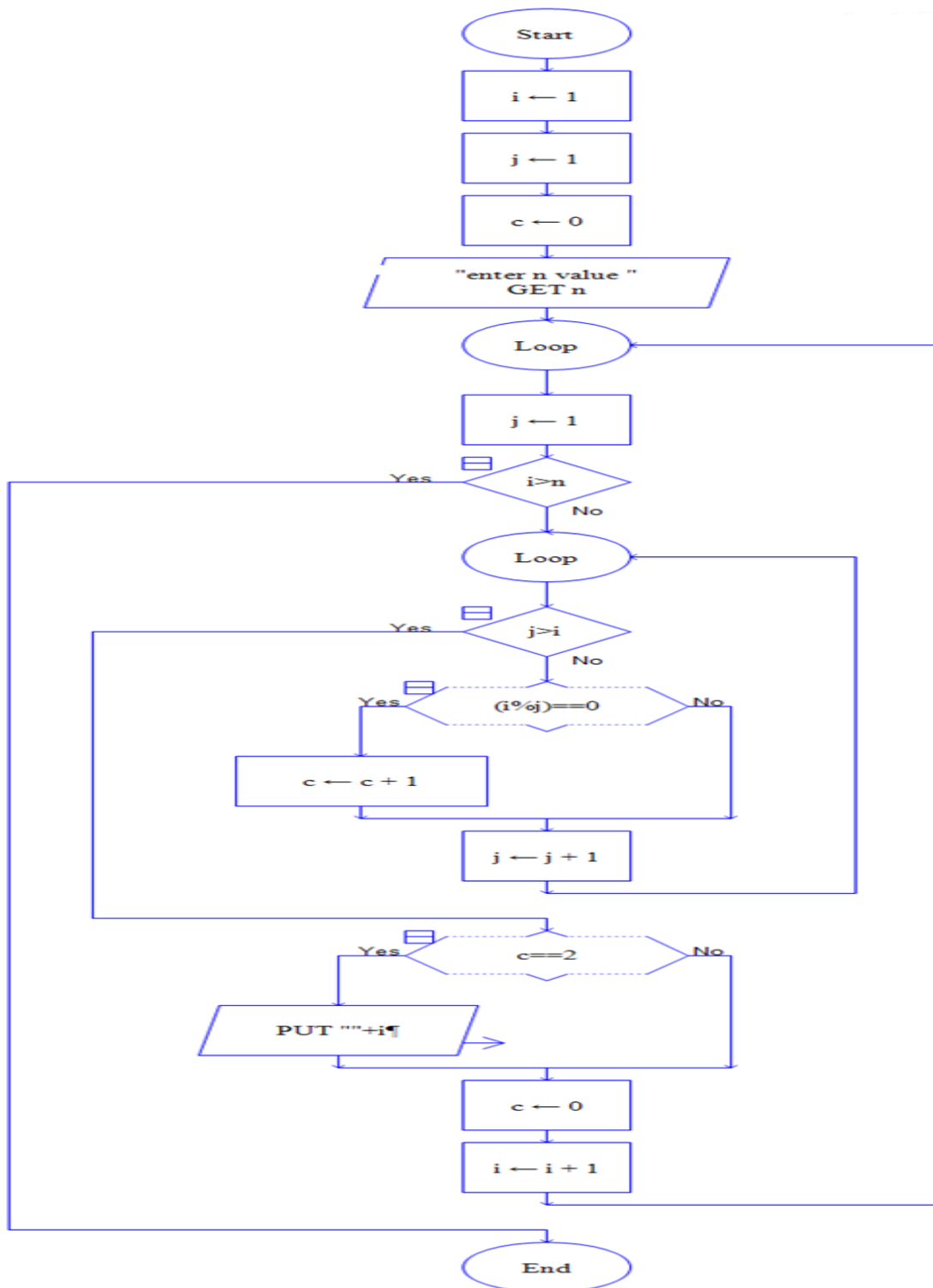
**Step 4:** In for loop check the condition if  $n \% i = 0$  then make flag = flag + 1

**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 d

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

If  $d > 0$  the roots will be distinct

If  $d == 0$  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

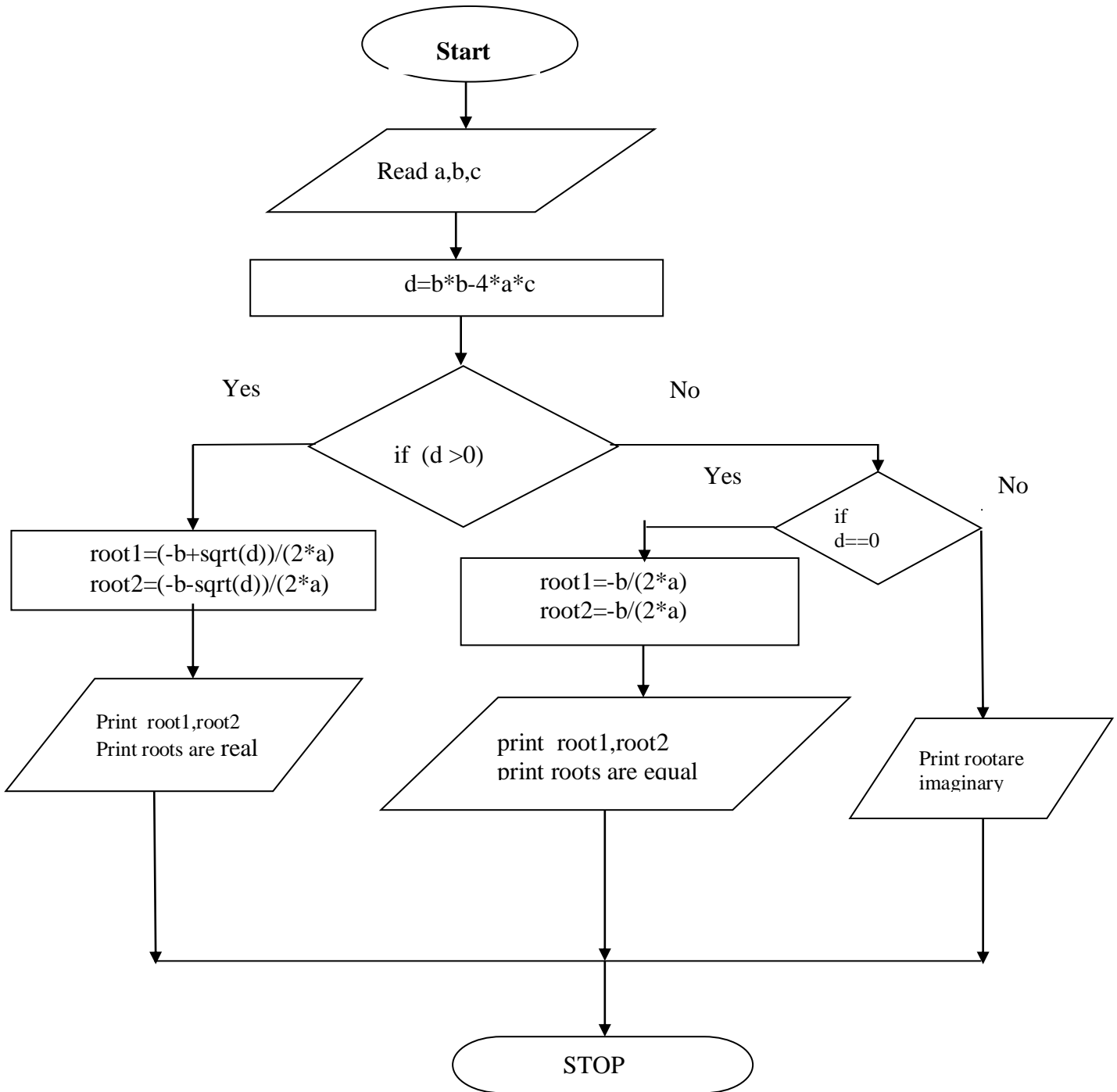
4.2.4: print roots are equal

else

(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

1. What is the output?

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

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

2. 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 :III Iterative Functions

### Demonstration

1. Input an integer (5 digits or fewer) containing only 0s and 1s (i.e., a “binary” integer) and print its decimal equivalent. [Hint: Use the remainder and division operators to pick off the “binary” number’s digits one at a time from right to left. Just as in the decimal number system, in which the rightmost digit has a positional value of 1, and the next digit left has a positional value of 10, then 100, then 1000, and so on, in the binary number system the rightmost digit has a positional value of 1, the next digit left has a positional value of 2, then 4, then 8, and so on. Thus the decimal number 234 can be interpreted as  $4 * 1 + 3 * 10 + 2 * 100$ . The decimal equivalent of binary 1101 is  $1 * 1 + 0 * 2 + 1 * 4 + 1 * 8$  or  $1 + 0 + 4 + 8$  or 13.]

```
#include<stdio.h>
int main(void)
{
int binary;
int number;
int decimal =0;
int highBit =16;
int factor =10000;
printf("%s", "Enter a binary number (5 digits maximum): ");
scanf("%d", &binary);
number = binary;
while(highBit >=1) {
decimal += binary / factor * highBit;
highBit /=2;
binary %= factor;
factor /=10;
```

```
printf("The decimal equivalent of %d is %d\n", number, decimal); } }
```

**Output :**

Enter a binary number ( 5 digits maximum ): 10111

Enter a binary number ( 5 digits maximum ): 10111

The decimal equivalent of 10111 is 16

The decimal equivalent of 10111 is 16

The decimal equivalent of 10111 is 20

The decimal equivalent of 10111 is 22

The decimal equivalent of 10111 is 23

**2. Armstrong numbers are numbers that are equal to the sum of their digits raised to power of the number of digits in them. The number 153, for example, equals  $1^3 + 5^3 + 3^3$ . Thus it is an Armstrong number. Write a program to display all three-digit Armstrong numbers.**

```
#include<stdio.h>

int main()
{
int n,r,sum=0,temp;
printf("enter the number=");
scanf("%d",&n);
temp=n;
while(n>0)
{
r=n%10;
sum=sum+(r*r*r);
n=n/10;
}
if(temp==sum)
printf("armstrong number ");
else
printf("not armstrong number");
return 0;
}
```

**Output:**

```
enter the number=153
armstrong number
enter the number=5
not armstrong number
```

**3. 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: 123456789

Total digits: 9

**4. 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
```

**5. Write a C program to calculate the following, where x is a fractional value.**

$$1 - \frac{x}{2} + \frac{x^2}{4} - \frac{x^3}{6} \dots\dots$$

```
#include <stdio.h>

Void main()
{
 float x,sum,no_row;
 int i,n;
 printf("Input the value of x :");
 scanf("%f",&x);
 printf("Input number of terms : ");
 scanf("%d",&n);
 sum=1; no_row =1;
 for(i=1;i<n;i++)
 {
 no_row = no_row*x/(float)i;
 sum=sum+ no_row;
 }
 printf("\nThe sum is : %f\n",sum);
}
```

**Output:**

Input the value of x :3

Input number of terms : 5

The sum is : 16.375000

**6. Write a C program to read in two numbers, x and n, and then compute the sum of this geometric progression:  $1+x+x^2+x^3+\dots+x^n$ . For example: if n is 3 and x is 5, then the program computes  $1+5+25+125$ .**

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{
int n, x, i, sum = 0;
printf("Enter the limit\n");
scanf("%d", &n);
printf("Enter the value of x\n");
scanf("%d", &x);
if(x < 0 || n < 0)
{
printf("illegal value");
}
else
{
for(i = 0; i <= n; i++)
sum=sum + pow(x, i);
}
printf("sum=%d", sum);
getch();
}
```

**Input & Output:**

Enter the limit



4

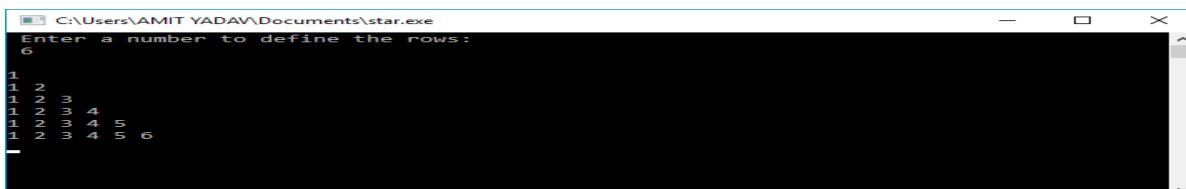
Enter the value of x

sum=31

**7. 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
```

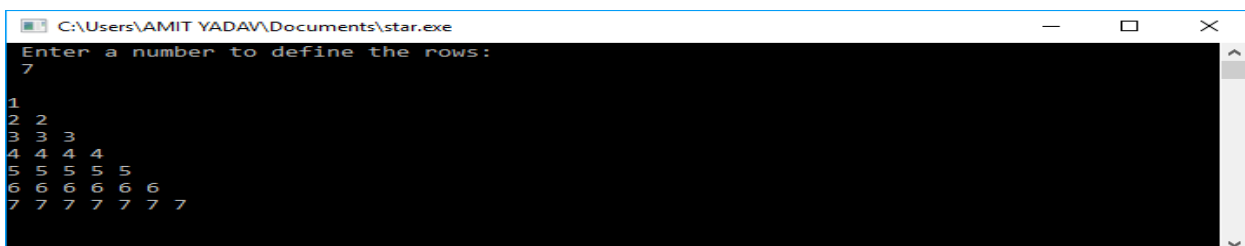
```
#include <stdio.h>
#include <conio.h>
void main()
{
 // declare the local variables
 int i, j, rows;
 printf (" Enter a number to define the rows: \n ");
 scanf("%d", &rows);
 printf("\n");
 for (i = 1; i <= rows; ++i)
 {
 for (j = 1; j <= i; ++j)
 {
 printf ("%d ", j);
 }
 printf ("\n");
 }
 getch();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



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

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 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 ", 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-6**  
**Experiment**

**Program 8:**

**8. Write a program that reads three nonzero integer values and determines and prints whether they could represent the sides of a triangle.**

**Problem Description:** Enter three nonzero integer values and determines and prints whether they could represent the sides of a triangle

**Algorithm:**

**Step 1:** start the program.

**Step 2:** take three sides as input.

**Step 3:** check if the sum of any two sides is equal to the third side.

**Step 4:** if false, a triangle cannot be formed, and stop the program.

**Step 5:** if true, check if all the three sides are equal.

**Step 6:** if yes, the triangle is an equilateral triangle.

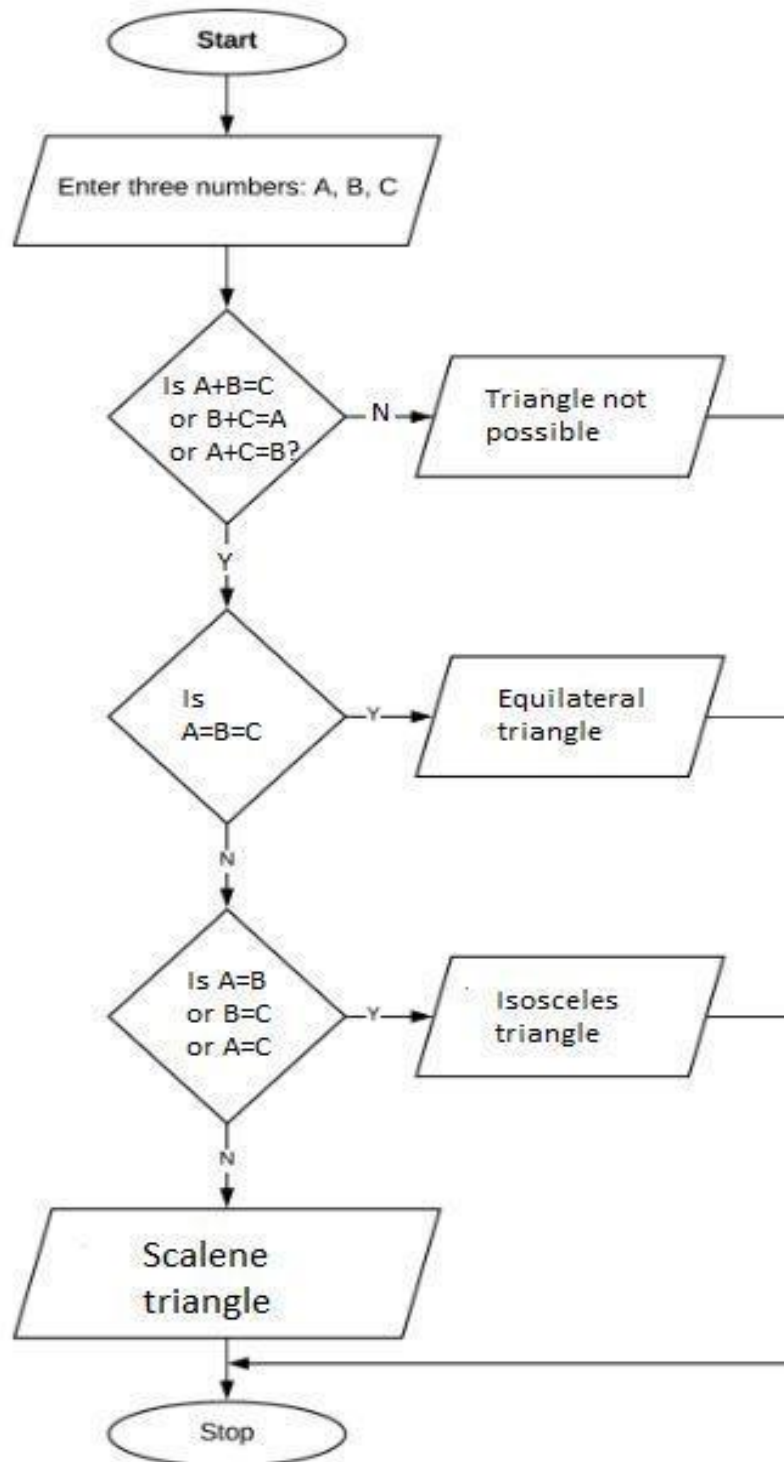
**Step 7:** if no, check whether any two of the three sides are equal.

**Step 8:** if yes, the triangle is an isosceles triangle.

**Step 9:** if no, the triangle is a scalene triangle.

**Step 10:** end the program

**Flow Chart:**



**Program 9:**

**9. Write a program that reads three nonzero integers and determines and prints whether they could be the sides of a right triangle.**

**Problem Description:** reads three nonzero integers and determines and prints whether they could be the sides of a right triangle.

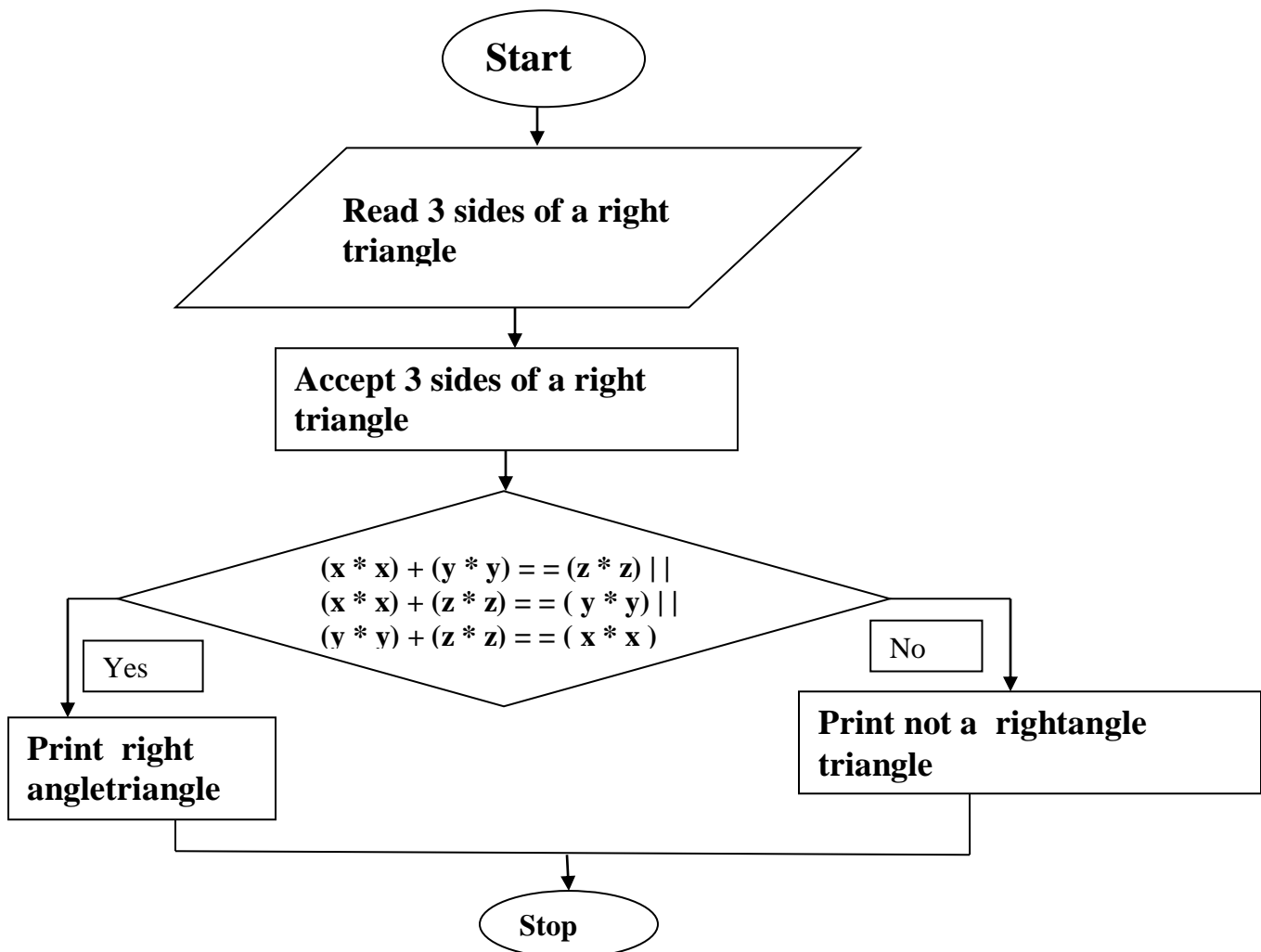
**Algorithm:**

**Step 1:** start

**Step 2:** take three sides as input.

**Step 3:** create the logic of the right-angle triangle with the help of pow ( $x^2+y^2=z^2$ ).

**Flowchart:**



## Program 10:

### 10. Write a C program to find the factorial of a positive integer.

**Problem Description:** N factorial is represented as N! where the exclamation point means factorial. For example,

$$1! = 1$$

$$2! = 1 * 2 = 2$$

$$3! = 1 * 2 * 3 = 6$$

$$4! = 1 * 2 * 3 * 4 = 24$$

...

$$N! = 1 * 2 * 3 * \dots * N$$

#### Algorithm:

**Step 1:** Start

**Step 2:** Declare variables n, factorial and i.

**Step 3:** Initialize variables

$$3.1: \text{factorial} = 1$$

$$3.2: i = 1$$

**Step 4:** Read value of n

**Step 5:** Repeat the steps until  $i = n$

$$5.1: \text{factorial} = \text{factorial} * i$$

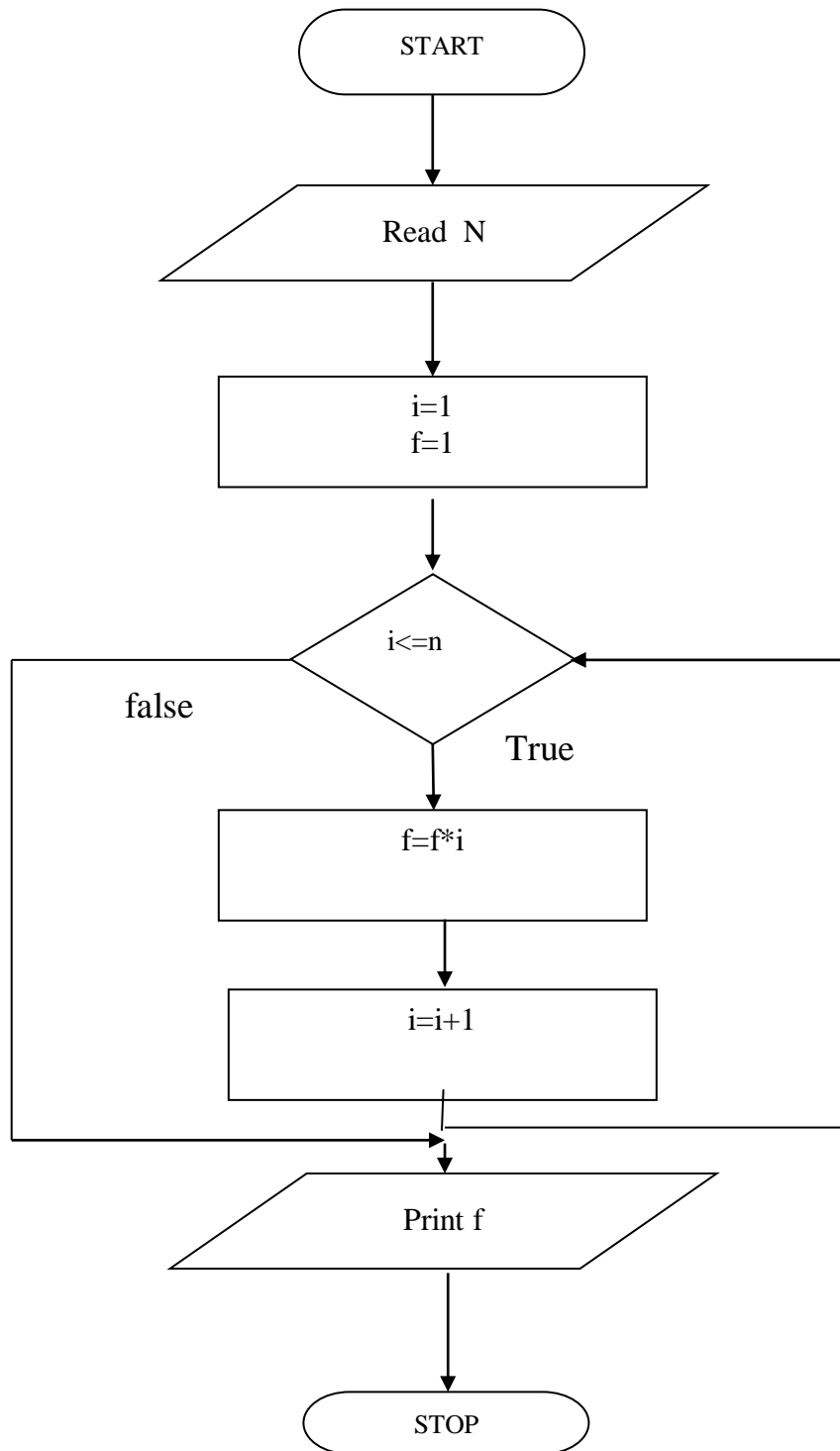
$$5.2: i = i + 1$$

**Step 6:** Display factorial

**Step 7:** Stop



**Flowchart:**



**Program 11:**

**11. Write a program that estimates the value of the mathematical constant  $e$  by using the formula:**

$$e^1 = 1 + \frac{1}{1!} + \frac{1^2}{2!} + \frac{1^3}{3!} + \dots$$

**Problem Description:** Estimates the value of the mathematical constant  $e$  by using the formula:

$$e^1 = 1 + \frac{1}{1!} + \frac{1^2}{2!} + \frac{1^3}{3!} + \dots$$

**Algorithm:**

**Step 1 :** Start

**Step 2 :** Read n

**Step 3:** initialize variables i=1, sum=1,no\_row=1, x=1;

**Step 4:** if (i<n)

**4.1 :**no\_row = no\_row \*x /(float)i ;

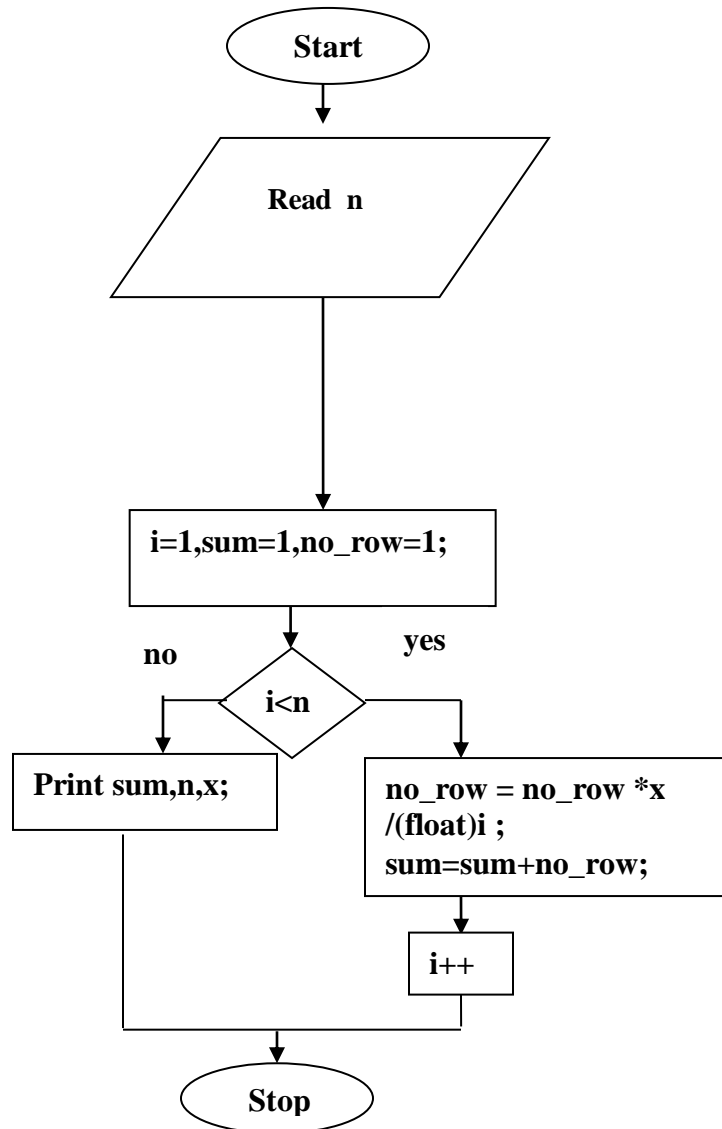
**4.2 :**sum=sum+no\_row;

**4.3 :**i++;

**Step 5:** Print sum,n,x;

**Step 6:** end;

**Flowchart:**



**Program 12:**

**12. Write a program that computes the value of  $e^x$  by using the formula  $e^x=1+x/1!+x^2/2!+x^3/3!+\dots,-\infty < x < \infty$**

**Problem Description:** computes the value of  $e^x$  by using the formula  $e^x=1+x/1!+x^2/2!+x^3/3!+\dots,-\infty < x < \infty$

**OUTPUT:**

Input the value of X=4

Number of terms=6

Sum is =42.866

**Algorithm:**

**Step-1:** Declare three variables float type and two as integer type

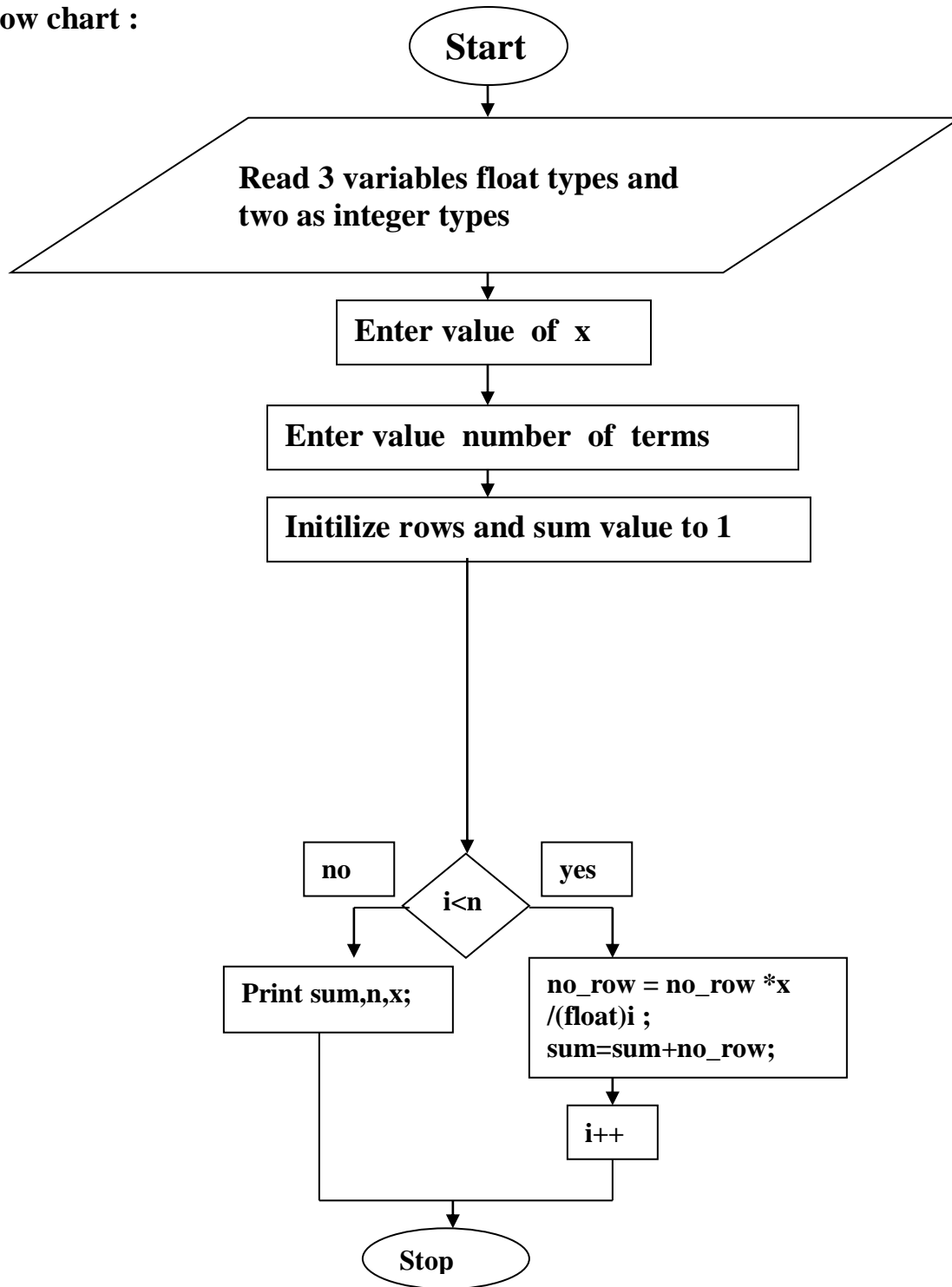
**Step -2:**Enter value for X

**Step-3:**Enter number of terms

**Step-4:**Intiliaze rows and sum value to 1

**Step-5:**USE this formula  $no\_rows=no\_rows*x/(float) i$  and  $sum=sum+no\_rows$ .

Flow chart :



## VIVA-QUESTIONS:

1. Which are not looping structures?

- (a) For loop
- (b) While loop
- (c) Do...while loop
- (d) **if...else**

2. The first expression in a for... loop is

- (a) Step value of loop
- (b) **Value of the counter variable**
- (c) Condition statement
- (d) None of the above

3. Which among the following is a unconditional control structure.

- (a) **Goto**
- (b) For
- (c) do-while
- (d) if-else

4. Continue statement

- (a) Breaks loop and goes to next statement after loop
- (b) **does not break loop but starts new iteration**
- (c) exits the program
- (d) Starts from beginning of program

5. The C code 'for( ; ; )' represents an infinite loop. It can be terminated by \_\_\_\_

- (a) **Break**
- (b) exit(0)
- (c) abort()
- (d) terminate

6. Which loop is guaranteed to execute at least one time?

- (a) for
- (b) while
- (c) do while
- (d) None of these

7. do-while loop terminates when conditional expression returns?

- (a) One
- (b) Zero
- (c) Non - zero
- (d) None of these

8. What is the output?

```
void main()
{
 while(true)
 {
 printf("Hello");
 break;
 }
}
```

- (a) Hello
- (b) Hello is printed infinite times
- (c) Blank
- (d) Error

## Week 7 :IV 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 `getresult(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 n 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()
{
 Inti,n;
 Floatstd_dev,sum=0,sumsqr=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;
 sumsqr=0;
 for(i=0;i<n;i++)
 {
 value=a[i]-mean;
 sumsqr=sumsqr+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

##### 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( $\leq 10$ ):

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: 2

3

Enter rows and column for second matrix: 3

2

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  $A^T$ .

**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

#### **4. Write C programs that use both recursive and non-recursive functions to find the factorial of a given integer.**

##### **Describe:**

Factorial of a positive integer (number) is the sum of multiplication of all the integers smaller than that positive integer. For example, factorial of 5 is  $5 * 4 * 3 * 2 * 1$  which equals to 120.

##### **Code:**

```
#include <stdio.h>
#include <conio.h>
void main()
{
int n, a, b;
 clrscr();
printf("Enter any number\n");
scanf("%d", &n);
 a = recfactorial(n);
printf("The factorial of a given number using recursion is %d \n", a);
 b = nonrecfactorial(n);
printf("The factorial of a given number using nonrecursion is %d ", b);
getch();
}
Int recfactorial(int x)
{
int f;
if(x == 0)
 {
return(1);
 }
}
```

```
else
{
 f = x * recfactorial(x - 1);
return(f);
}
}
Int nonrecfactorial(int x)
{
int i, f = 1;
for(i = 1; i <= x; i++)
{
 f = f * i;
}
return(f);
}
```

**Output :**

Enter any number5

The factorial of a given number using recursion is 120

The factorial of a given number using nonrecursion is 120

**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  $i \leq n2$

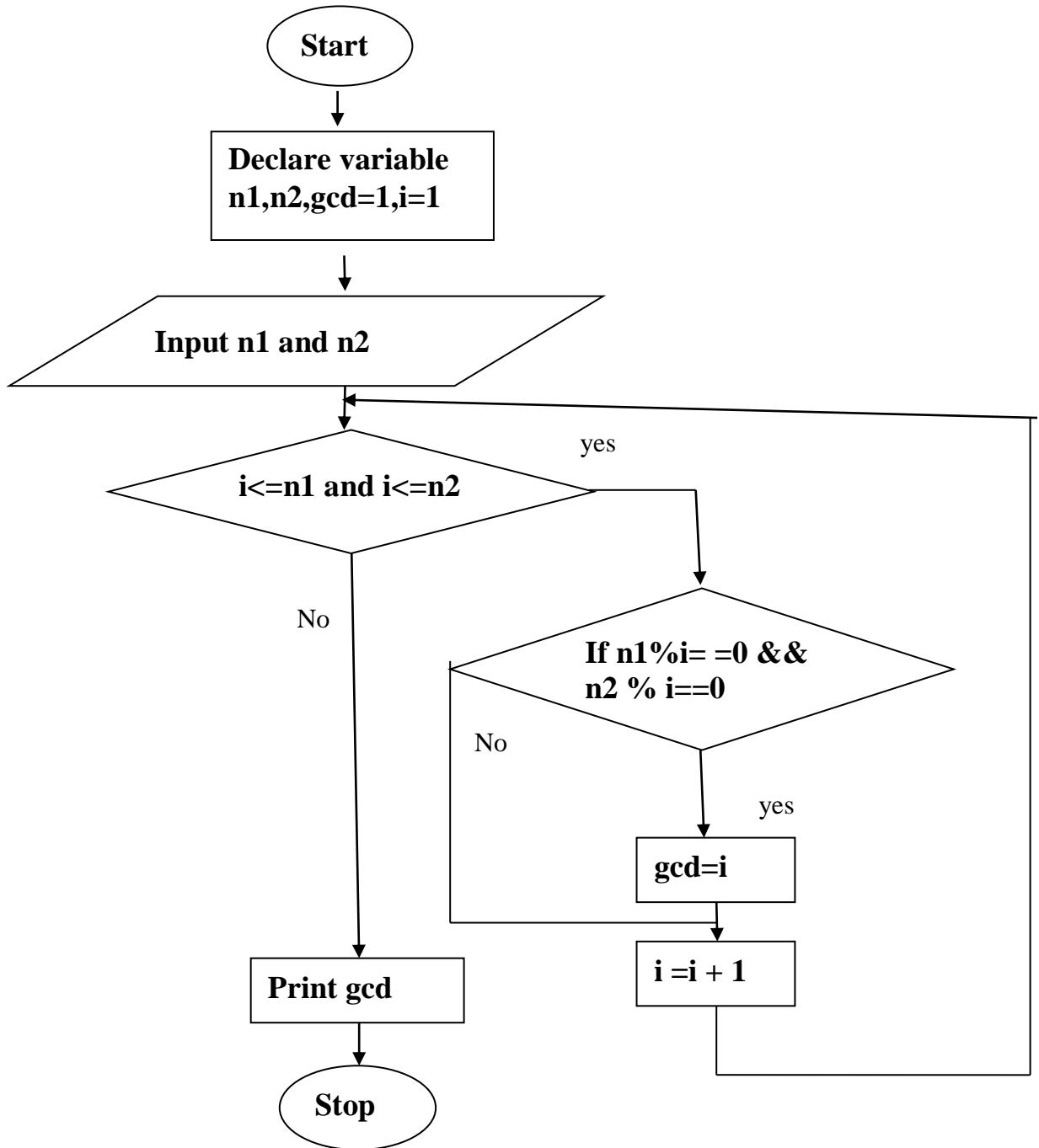
**Step 4.1:** If  $n1 \% i == 0 \ \&\& \ n2 \% i == 0$ :

**Step 4.2:** gcd = i

**Step 5:** Print gcd

**Step 6:** Stop

**Flow chart :**





## Program 7:

### 7. 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  $i \leq Y$

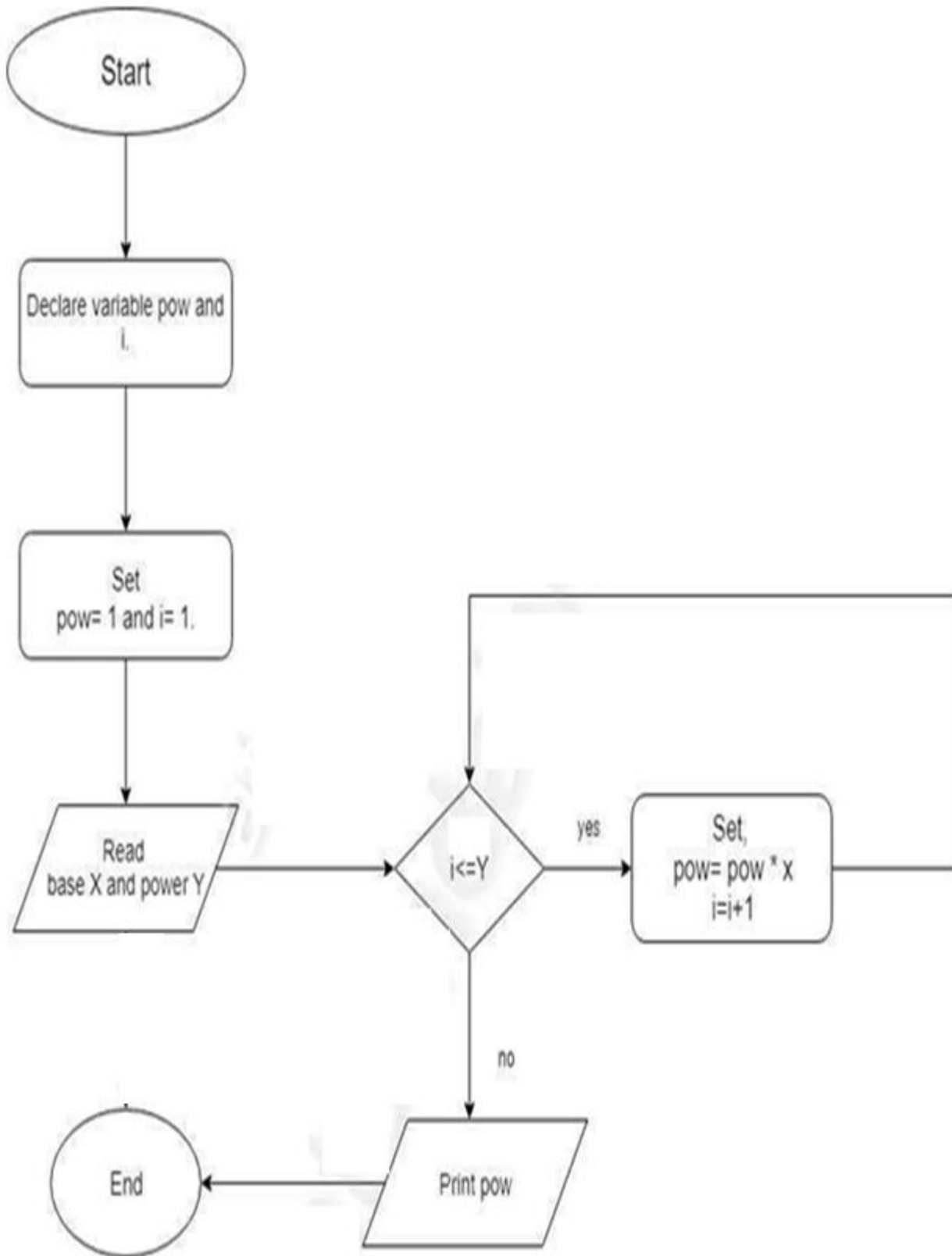
**5.1:** Set,  $pow = pow * x$

**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:



## Program 8:

**8. Write a program for reading elements using a pointer into an array and display the values using the array.**

### Problem Description:

If we are entering 5 elements ( $N = 5$ ), with array element values as 4, 9, 10, 56 and 100 then, Sum of Elements of the array will be:  $4 + 9 + 10 + 56 + 100 = 179$

### Algorithm:

**Step1:** start

**Step2:** Read array a[] with n elements

**Step 3:** initialize pointer  $p = \&a[0]$  [or  $p = a$ ]

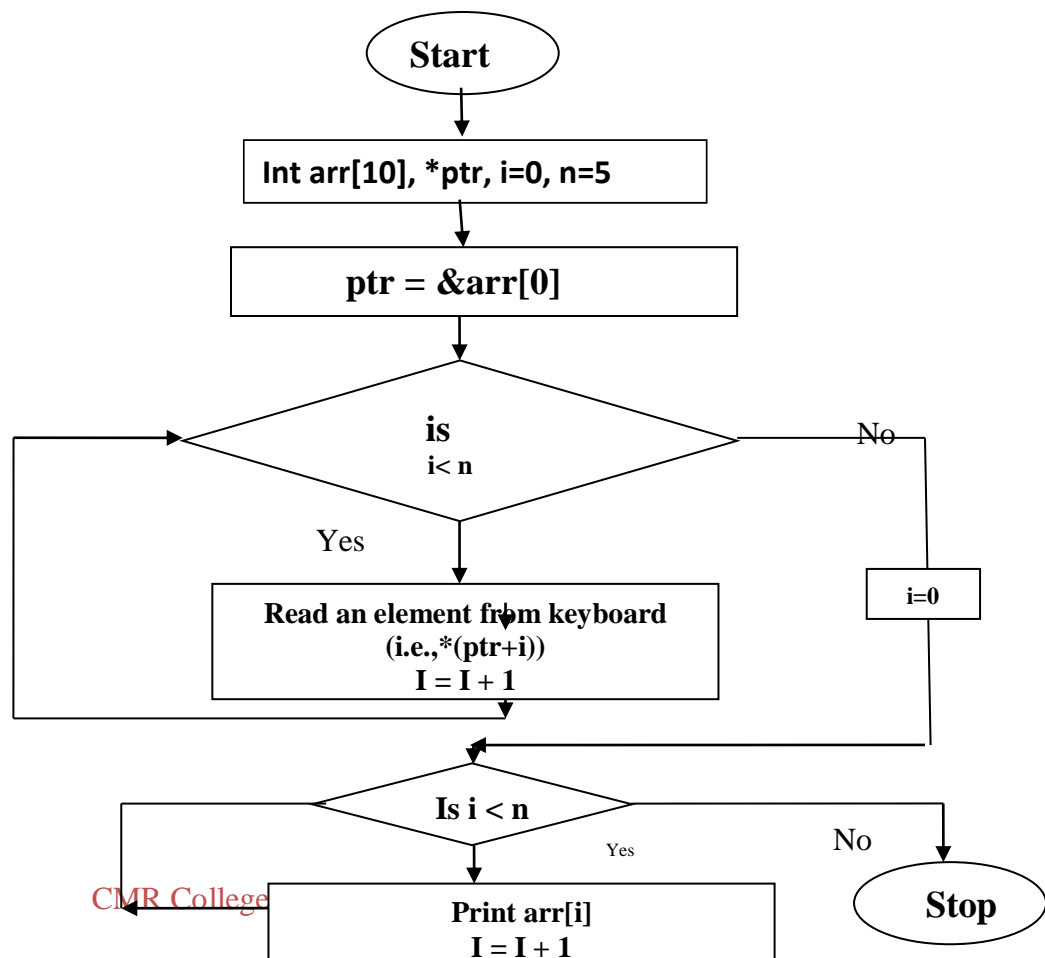
**Step 4:** if  $i < n$  go to next step otherwise go to step 7

**Step 5:** print  $*(p+i)$

**Step 6:**  $i = i + 1$  go to step 4

**Step 7:** stop

### Flow Chart :



## Program 9:

**9. Write a program for display values reverse order from an array using a pointer.**

**Problem Description:** When using array name itself as a pointer, note that array name always holds the address of first element in the array. In the following program, array name arr is equivalent to &arr[0]. So array name is considered as indirect pointer.

### Algorithm:

**Step1:** start

**Step2:** Read array a[] with n elements

**Step 3:** initialize pointer  $p = \&a[n-1]$  and  $i = 0$

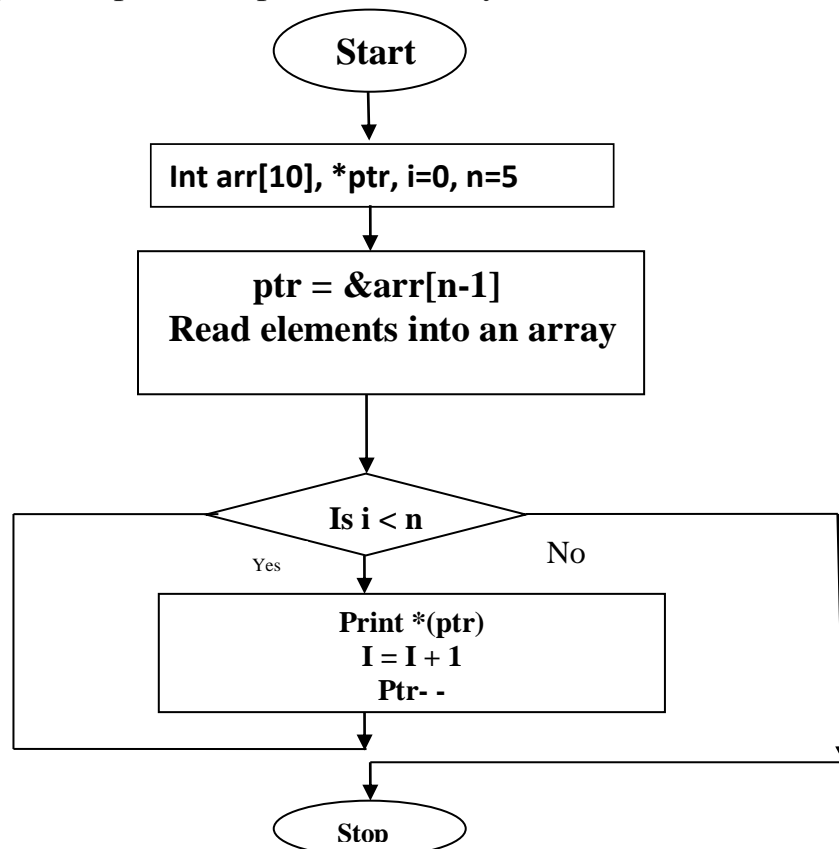
**Step 4:** if  $i < n$  go to next step otherwise go to step 7

**Step 5:** print  $*(p)$

**Step 6:**  $p--$  and  $i=i+1$  go to step 4 // p decrements by its scale factor

**Step 7:** stop

### Flow Chart :



### Program 10:

**10. Write a program through a pointer variable to sum of n elements from an array.**

**Problem Description:** We have to write a program in C such that it calculates the sum of elements of an array using pointers. The program should dynamically allocate a piece of memory for that array and use a pointer to point to that array memory as well as traverse that array using a pointer.

#### Algorithm:

**Step 1:** Declare the integer array

**Step 2:** Take input the elements of the integer array

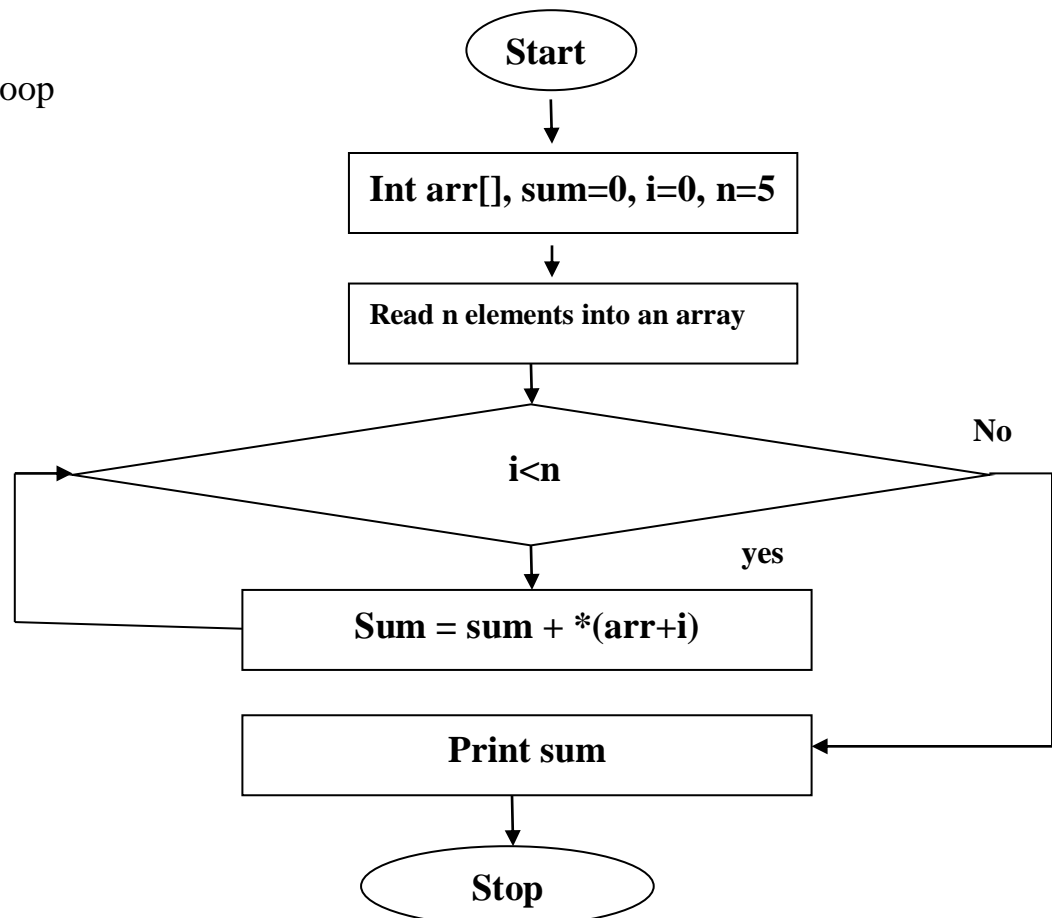
**Step 3:** Initialize pointer i to 0

**Step 4:** for(i=0 ; i<n ; i++) //here n is the length of the integer array

**Step 5:** print a[i]

**Step 6:** End of for loop

#### 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 9 : V Files Demonstration

**1. Write a C program to display the contents of a file to standard output device.**

**Problem Description:** Create a file, and open it write something as text in write mode and specify ctrl + z as the end of the text, then close file. Display the content of the file in read mode.

**Program:**

```
void main()
{
 FILE *fp1;
 char ch;
 clrscr ();
 fp1 = fopen ("file1.txt", "w");
 printf ("enter text press ctrl+z at the end");
 while ((ch = getchar ()) != EOF)
 {
 putc(ch, fp1);
 }
 fclose (fp1);
 fp1 =fopen ("file1.txt", "r");
 ch=' ';
 while ((ch = getc (fp1)) != EOF)
 {
 Printf("%c",ch);
 }
 fclose (fp1);
 getch ();
}
```



**Input:**

enter text press ctrl+z at the end

Hello

How r u.

**Output:**

File2 contents are

Hello

How r u

**2. Write a C program which copies one file to another, replacing all lowercase characters with their uppercase equivalents**

**Problem Description:** Read the string with all lowercase, replace with equivalent uppercase text.

For example: If A file with text c programming its copied into another file with text CPROGRAMMING.

**Program:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
 char ch;
 FILE *fp1,*fp2;
 fp1=fopen("asc.txt","r");
 fp2=fopen("b.txt","w");
 while(!feof(fp1))
 {
 ch=fgetc(fp1);
 if(islower(ch))
 ch=toupper(ch);
 fputc(ch,fp2);
 }
 fclose(fp1);
 fclose(fp2);
 fp2=fopen("b.txt","r");
 while(!feof(fp2))
 {
 ch=fgetc(fp2);
 printf("%c",ch);
 }
}
```

```
}
fclose(fp2);
}
```

**Output:**

All the lower-case letters in source.txt are converted into their upper case form in following target.txt file.

**3. Write a C program to count the number of times a character occurs in a text file. The file name and the character are supplied as command line arguments.  
program:**

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define BUFFER_SIZE 1000

/* Function declarations */

int countOccurrences(FILE *fptr, const char *word);

int main()

{

 FILE *fptr;

 char path[100];

 char word[50];

 int wCount;

 /* Input file path */

 printf("Enter file path: ");

 scanf("%s", path);

 /* Input word to search in file */

 printf("Enter word to search in file: ");

 scanf("%s", word);

 /* Try to open file */

 fptr = fopen(path, "r");
```

```

 /* Exit if file not opened successfully */
if (fptr == NULL)
 {
printf("Unable to open file.\n");
printf("Please check you have read/write previleges.\n");
exit(EXIT_FAILURE);
 }

 // Call function to count all occurrence of word
wCount = countOccurrences(fptr, word);
printf("%s' is found %d times in file.", word, wCount);

 // Close file
fclose(fptr);
return 0;
}

/**
 * Returns total occurrences of a word in given file.
 */
int countOccurrences(FILE *fptr, const char *word)
{
char str[BUFFER_SIZE];
char *pos;
int index, count;
count = 0;

```

```
// Read line from file till end of file.
while ((fgets(str, BUFFER_SIZE, fptr)) != NULL)
{
index = 0;

// Find next occurrence of word in str
while ((pos = strstr(str + index, word)) != NULL)
{ // Index of word in str is

// Memory address of pos - memory // address of str.
index = (pos - str) + 1;
count++;
}
}
return count;
}
```

**Output:**

Enter file path: data/file3.txt

Enter word to search in file: Codeforwin

'Codeforwin' is found 2 times in file.

## WEEK-10 EXPERIMENT

### Program 4:

#### 4. Write a C program that does the following:

It should first create a binary file and store 10 integers, where the file name and 10 values are given in the command line. (hint: convert the strings using atoi function) Now the program asks for an index and a value from the user and the value at that index should be changed to the new value in the file. (hint: use fseek function) The program should then read all 10 values and print them back

**Problem Description:** If you have many records inside a file and need to access a record at a specific position, you need to loop through all the records before it to get the record. This will waste a lot of memory and operation time. An easier way to get to the required data can be achieved using `fseek()`.

As the name suggests, `fseek()` seeks the cursor to the given record in the file. The first parameter stream is the pointer to the file. The second parameter is the position of the record to be found, and the third parameter specifies the location where the offset starts. This program will start reading the records from the file `program.bin` in the reverse order (last to first) and prints it.

#### Algorithm:

**Step 1:** start the program.

**Step 2:** take ten integers as input, \*fptr and accept ten integers.

**Step 3:** check if (fptr=fopen==NULL)

**Step 4:** if true, print “errors opening file”

**Step 5:** if false, use `fseek(ptr,10 integers,seek_end)`

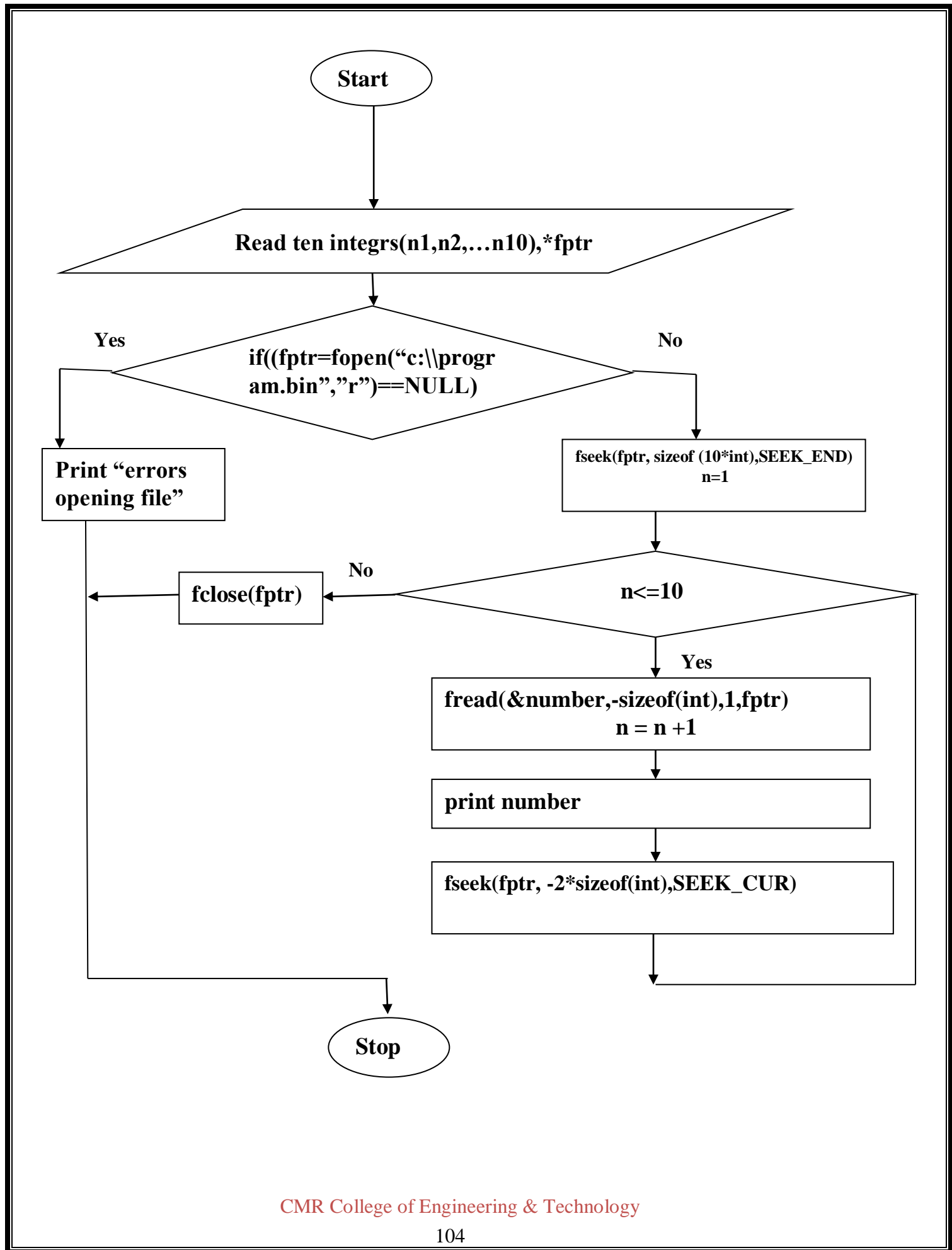
**Step 6:** if number is larger then 10

**Step 7:** if no, close ptr

**Step 8:** if yes, use `fread(10 integers)` and print 10 files

**Step 9:** use `fseek(seek_cur)`

**Step 10:** end the program





### **Program 5:**

**5. Write a C program to merge two files into a third file (i.e., the contents of the first file followed by those of the second are put in the third file).**

**Problem Description:** C program to merge two files and store their contents in another file. The files to be merged are opened in "read" mode and the file that contains contents of both the files is opened in "write" mode. For merging, we open a file and read it character by character and store the read contents in the merged file then repeat this for the second file. The files are read until their EOF (end of file) is reached.

### **Algorithm :**

**Step 1.:**Open file1.txt and file2.txt in read mode.

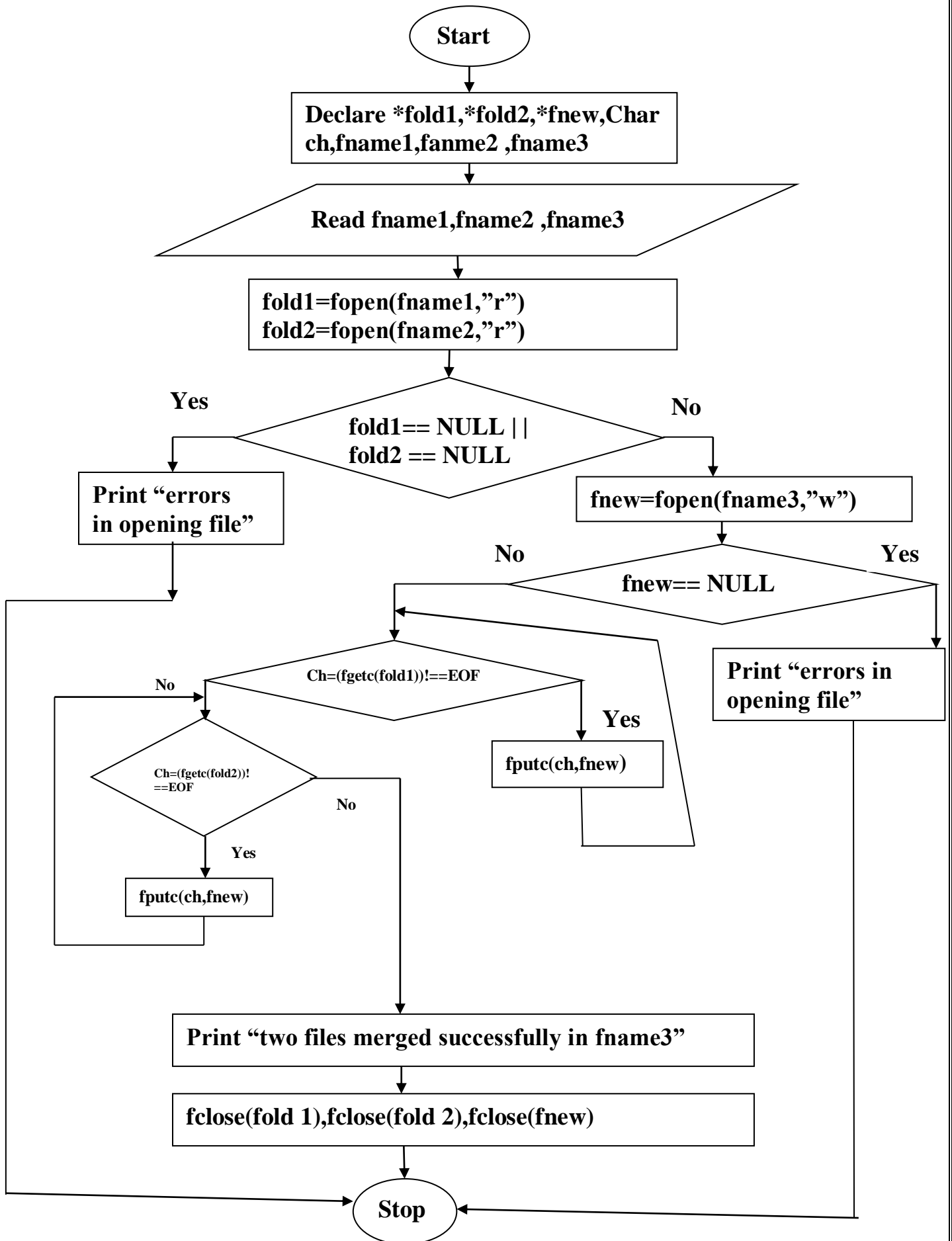
**Step 2.:** Open file3.txt in write mode.

**Step 3.:** Run a loop to one by one copy characters of file1.txt to file3.txt.

**Step 4.:** Run a loop to one by one copy characters of file2.txt to file3.txt.

**Step 5.:** Close all files.

### **Flowchart :**



## VIVA-QUESTIONS:

1. What is the keyword used to declare a C file pointer.?
  - A. file
  - B. FILE
  - C. FILEFP
  - D. filefp
  
2. Which one of the following is correct syntax for opening a file.
  - A. FILE \*fopen(const \*filename, const char \*mode)
  - B. FILE \*fopen(const \*filename)
  - C. FILE \*open(const \*filename, const char \*mode)
  - D. FILE open(const\*filename)
  
3. A mode which is used to open an existing file for both reading and writing \_\_\_\_\_
  - A. "W"
  - B. "W+"
  - C. "R+"
  - D. "A+"
  
4. A data of the file is stored in ...
  - A. Ram
  - B. Hard disk
  - C. Rom
  - D. None
  
5. getc() returns EOF when
  - A. End of files is reached
  - B. When getc() fails to read a character
  - C. Both of the above
  - D. None of the above
  
6. Select a function which is used to write a string to a file \_\_\_\_\_
  - A. pits()
  - B. putc()
  - C. fputs()
  - D. fgets()
  
7. Where is a file temporarily stored before read or write operation in C language.?
  - A. Notepad

- B. RAM
- C. Hard disk
- D. Buffer

8. Select text file in which data is stored in \_\_\_\_\_

- A. Binary code
- B. Octal code
- C. text code
- D. ASCII code

9. Which function will return the current file position for stream?

- A. fgetpos()
- B. ftell()
- C. fsetpos()
- D. fseek()

10. What type of array is generally generated in Command-line argument?

- A. Jagged Array
- B. 2-Dimensional Rectangular Array
- C. Single dimension array
- D. 2-Dimensional Square Array

## Week 11 : VI Strings

### Demonstration

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

**Describe:**

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

| Symbol | Value |
|--------|-------|
|--------|-------|

|   |      |
|---|------|
| I | 1    |
| V | 5    |
| X | 10   |
| L | 50   |
| C | 100  |
| D | 500  |
| M | 1000 |

**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, C, D, M): \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 'T': value = 1;
break;
case 'V': value = 5;
break;
case 'X': value = 10;
break;
case 'L': value = 50;
break;
case 'C': value = 100;
break;
case 'D': value = 500;
break;
case 'M': value = 1000;
break;
case '\0': value = 0;
break;
default: value = -1;
}
return value;
}
```

**Output :**

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

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, L, C, D, and M stand respectively for 1, 5, 10, 50, 100, 500, and 1,000 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

---

|    |       |
|----|-------|
| 1  | I     |
| 2  | II    |
| 3  | III   |
| 4  | IV    |
| 5  | V     |
| 6  | VI    |
| 7  | VII   |
| 8  | VIII  |
| 9  | IX    |
| 10 | X     |
| 11 | XI    |
| 12 | XII   |
| 13 | XIII  |
| 14 | XIV   |
| 15 | XV    |
| 16 | XVI   |
| 17 | XVII  |
| 18 | XVIII |
| 19 | XIX   |
| 20 | XX    |
| 21 | XXI   |
| 22 | XXII  |

|    |         |
|----|---------|
| 23 | XXIII   |
| 24 | XXIV    |
| 25 | XXV     |
| 26 | XXVI    |
| 27 | XXVII   |
| 28 | XXVIII  |
| 29 | XXIX    |
| 30 | XXX     |
| 31 | XXXI    |
| 32 | XXXII   |
| 33 | XXXIII  |
| 34 | XXXIV   |
| 35 | XXXV    |
| 36 | XXXVI   |
| 37 | XXXVII  |
| 38 | XXXVIII |
| 39 | XXXIX   |
| 40 | XL      |
| 41 | XLI     |
| 42 | XLII    |
| 43 | XLIII   |
| 44 | XLIV    |
| 45 | XLV     |
| 46 | XLVI    |
| 47 | XLVII   |
| 48 | XLVIII  |
| 49 | XLIX    |
| 50 | L       |

**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 insert

programming

Enter the position to insert

10

computerprogrammingis 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 delete5

Enter the number of characters to be deleted8

compgood



## 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:

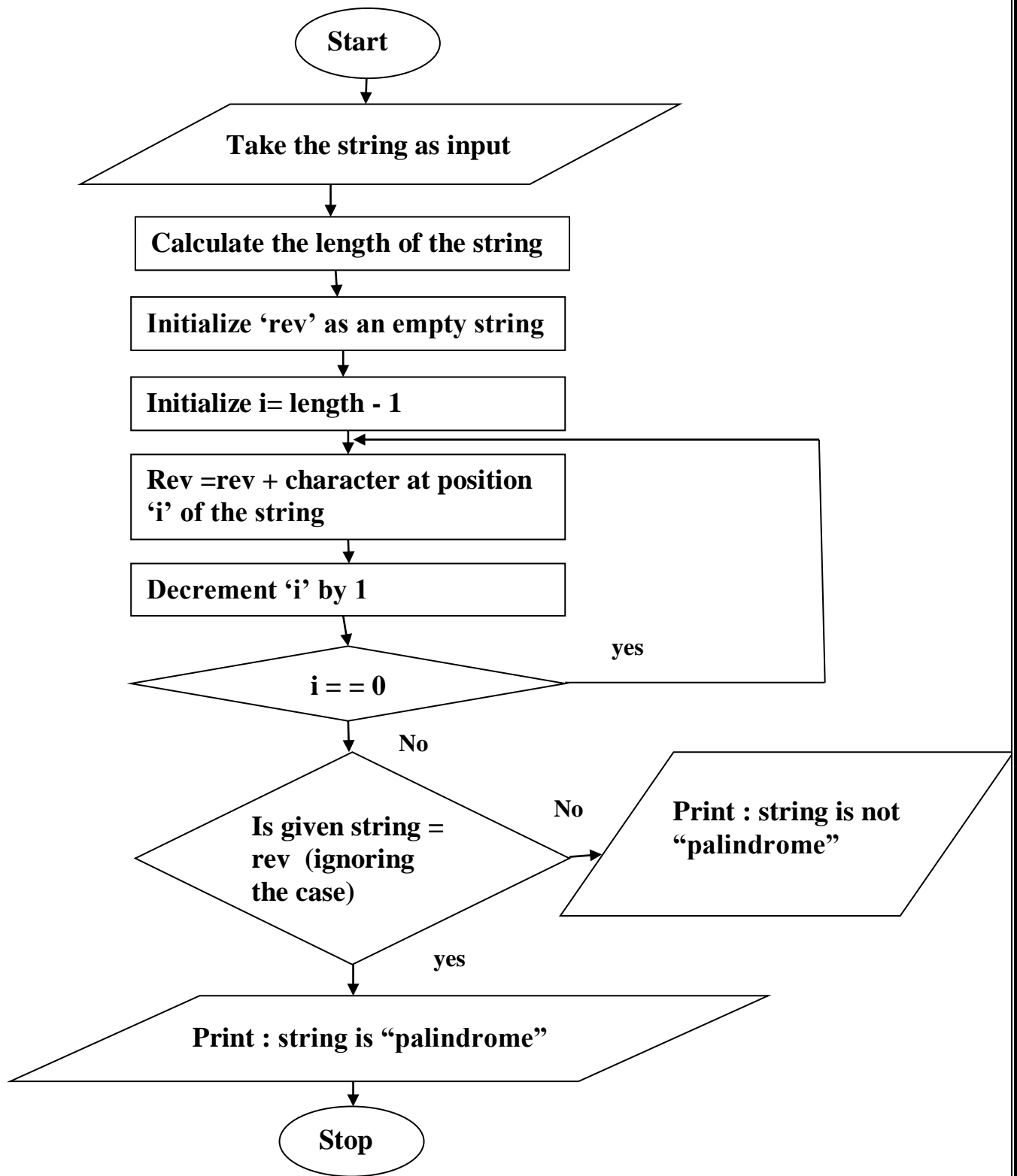
**7.1:** Print “Given string is palindrome”

**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)

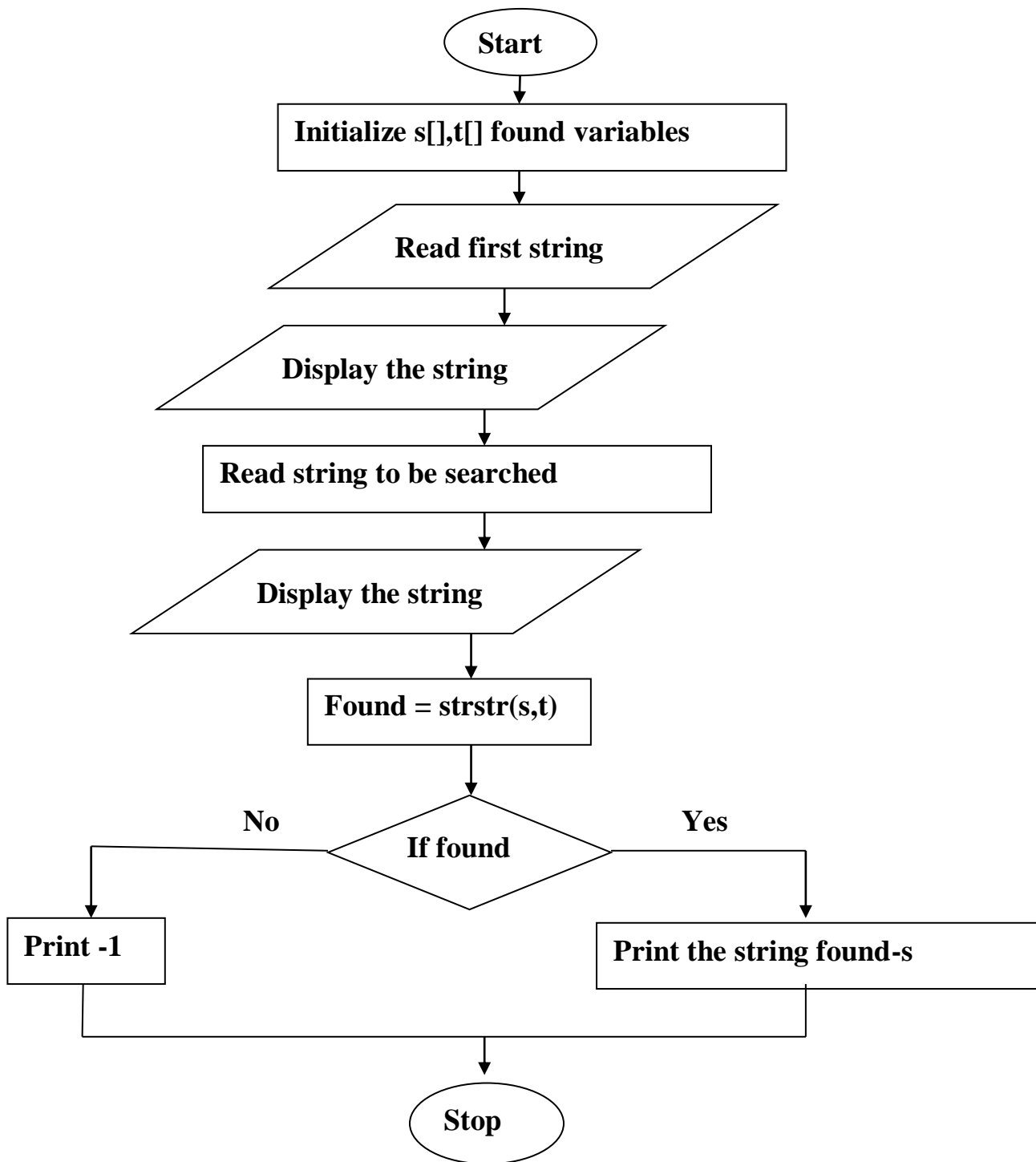
ii. if found print the second string is found in the first string at the position. If

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.

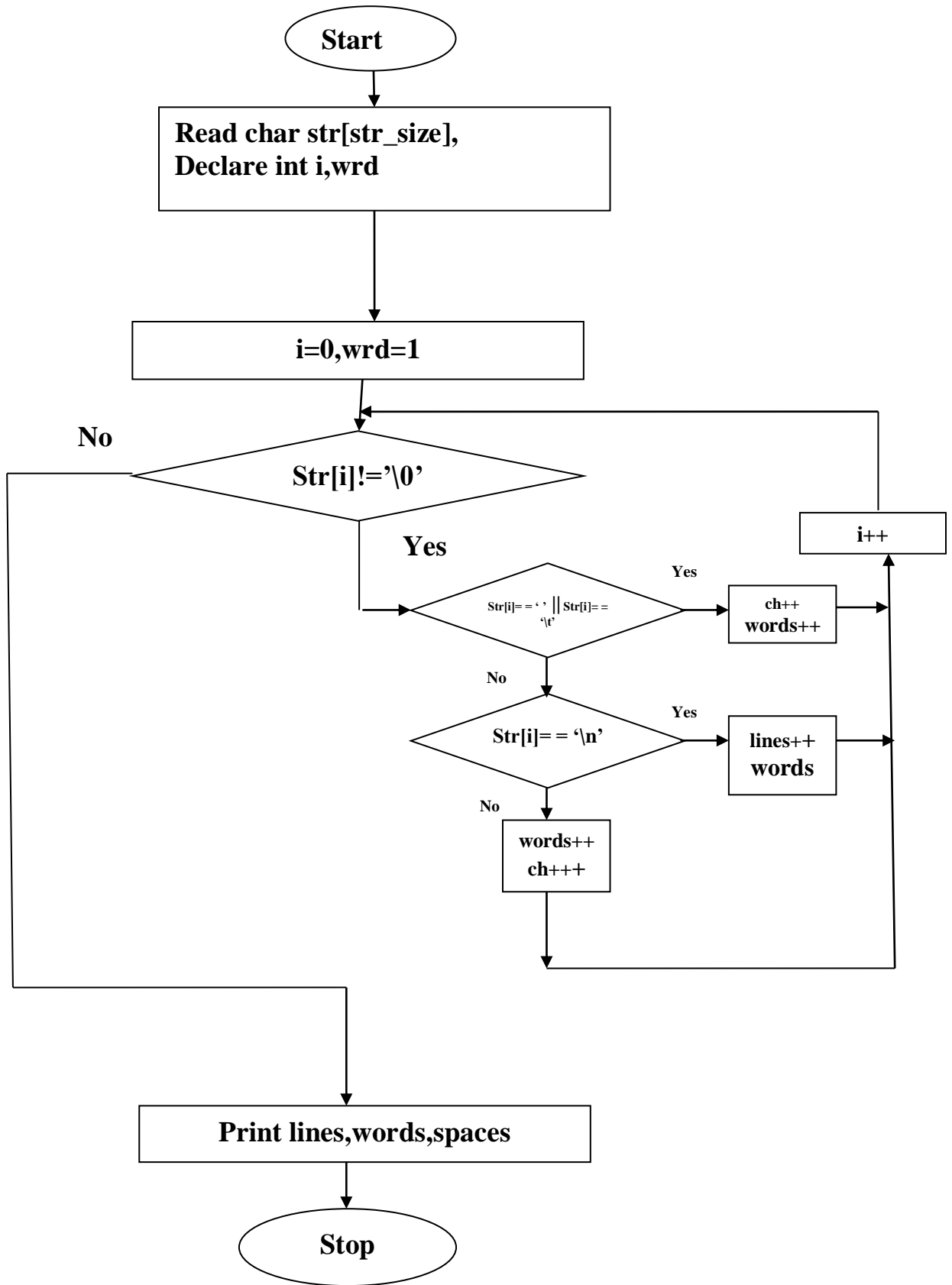
**Step 5:** Using for loop search for characters except space and new line in the string and consecutively increment a variable count for characters.

**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. strcpy()
- 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

**Week 13 Sorting and Searching:**  
**Demonstration**

**1 . Write a C program that uses nonrecursive function to search for a Key value in a given List of integers using line ar search method.**

**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 nonrecursive function to search for a Key value in a given Sorted list of integers using binary search method.**

```
#include <stdio.h>
int BinarySearching(int arr[], int max, int element)
{
int low = 0, high = max - 1, middle;
while(low <= high)
{
middle = (low + high) / 2;
if(element > arr[middle])
low = middle + 1;
else if(element < arr[middle])
high = middle - 1;
else
return middle;
}
return -1;
}
int main()
{
int count, element, limit, arr[50], position;
printf("Enter the Limit of Elements in Array:\t");
scanf("%d", &limit);
printf("Enter %d Elements in Array: \n", limit);
for(count = 0; count < limit; count++)
{
scanf("%d", &arr[count]);
```

```
}
printf("Enter Element To Search:\t");
scanf("%d", &element);
position = BinarySearching(arr, limit, element);
if(position == -1)
{
printf("Element %d Not Found\n", element);
}
else
{
printf("Element %d Found at Position %d\n", element, position + 1);
}
return 0;
}
```

### OUTPUT:

```
Enter the Limit of Elements in Array:
5
Enter 5 Elements in Array:
```

```
93
25
57
76
Enter Element to Search: 76
Element 76 Found at Position 5
```

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

**PROGRAM:**

```
#include<stdio.h>

#include<conio.h>

void main()

{

int n, a[20], temp, i, j;

clrscr();

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

scanf("%d", &n);

printf("Enter the array elements\n");

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

{

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

}

for(i = 0; i < n - 1; i++)

{

for(j = 0; j < n - 1; j++)

{

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

{

temp = a[j];

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

a[j + 1] = temp;

}

}

}

}
```

```
printf("The sorted array is\n");
for(i = 0; i < n; i++)
printf("%d\n", a[i]);
getch();
}
```

**Output:**

Enter the size of the array: 5

Enter the array elements:50 40 30 20 10

The sorted array is: 10 20 30 40 50

## WEEK-14 EXPERIMENT

### Program 4:

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

**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 (for Ascending order), then these two are swapped.

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

### 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.



15 > 20  
FALSE



15 > 10  
TRUE  
SWAP



10 > 30  
FALSE



10 > 50  
FALSE



10 > 18  
FALSE



10 > 5  
TRUE  
SWAP



5 > 45  
FALSE

List after 1st iteration



**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.

List after 2nd iteration



**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.

List after 3rd iteration



**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.

List after 4th iteration



**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.

List after 5th iteration



**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.

List after 6th iteration



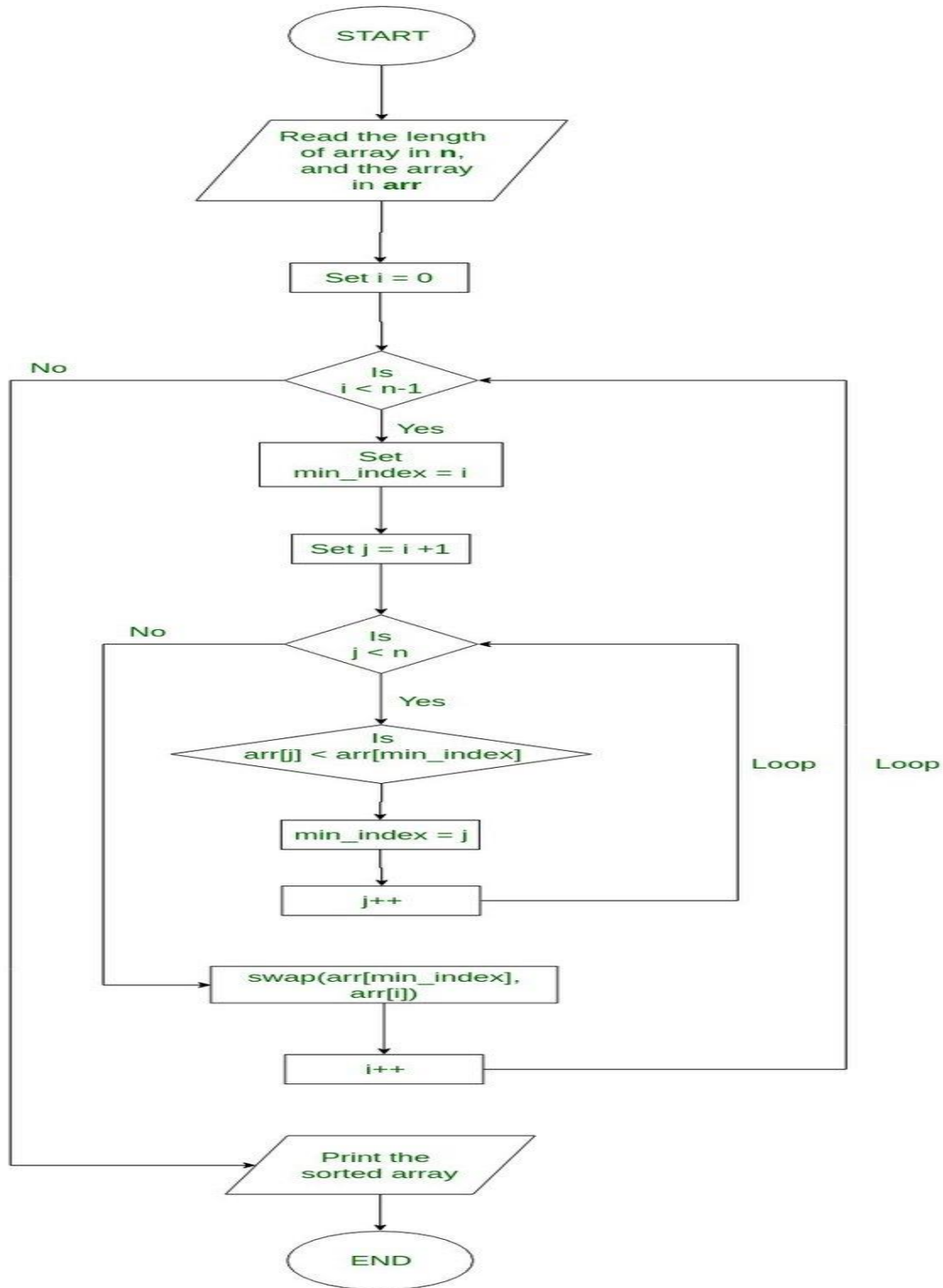
**Iteration #7**

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.

List after 7th iteration



Final sorted list



**Flowchart for Selection Sort**



## Program 5:

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

**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 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 insert 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 element, 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       |

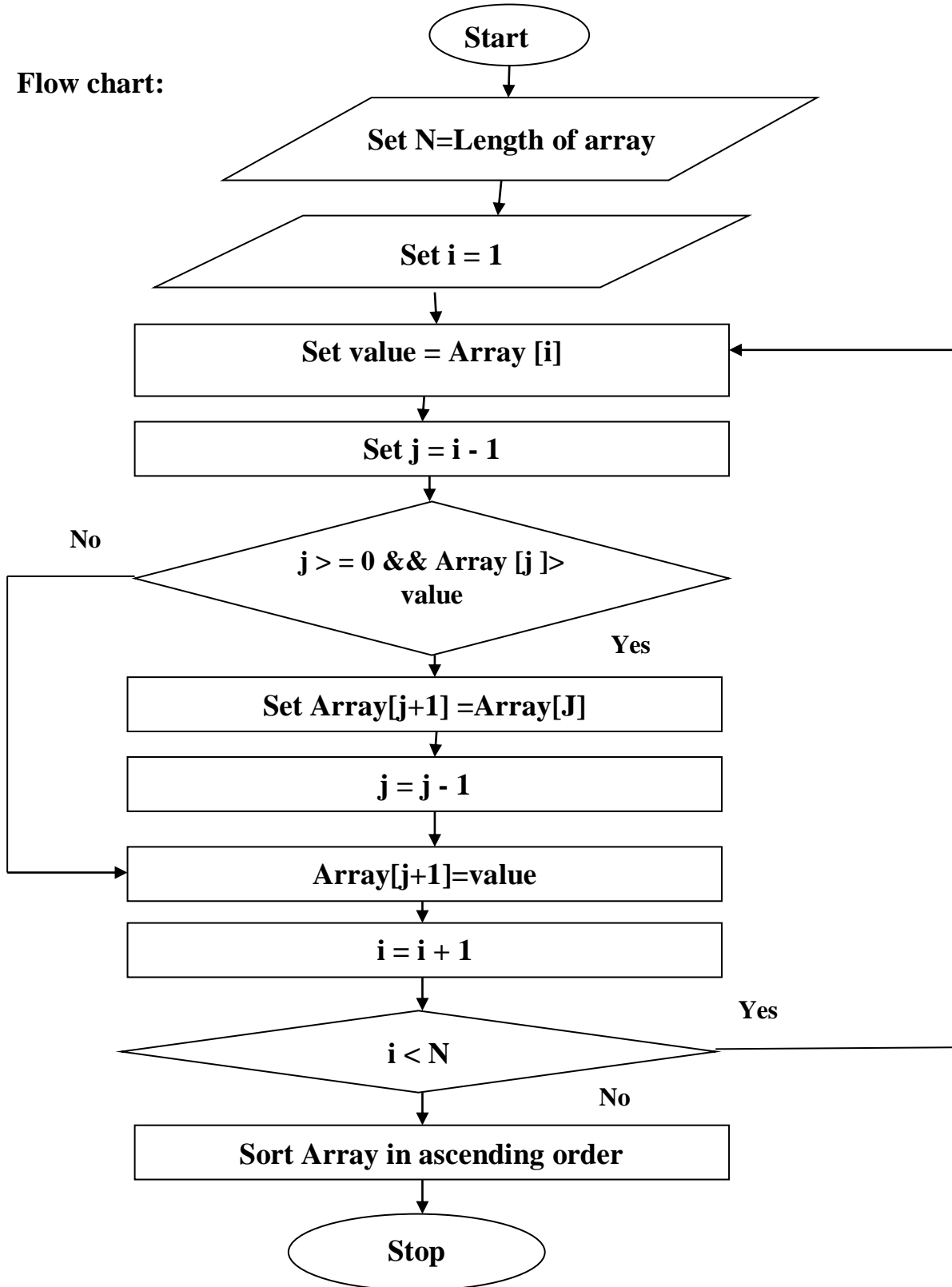
To move element 45 from unsorted to sorted portion, Compare 45 with 50 and 30. Since 45 is larger than 30, move 50 one position to the right in the list and insert 45 after 30 in the sorted list.

| 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 |
|---|----|----|----|----|----|----|----|

Flow chart:



## Program 6:

### 6. Write a C program that sorts a given array of names.

#### Problem Description:

In this program, we need to sort the given array in ascending order such that elements will be arranged from smallest to largest. This can be achieved through two loops. The outer loop will select an element, and inner loop allows us to compare selected element with rest of the elements.

#### Algorithm :

**Step 1:** Create an array of string and initialize it with the values.

**Step 2:** For loop from  $i=0$  to  $i<\text{size\_of\_array}$ :

A. Nest another for loop from  $j=0$  to  $j<\text{size\_of\_array}-i-1$ :

i. Check if( $\text{strcmp}(\text{array}[j], \text{array}[j+1]) > 0$ ):

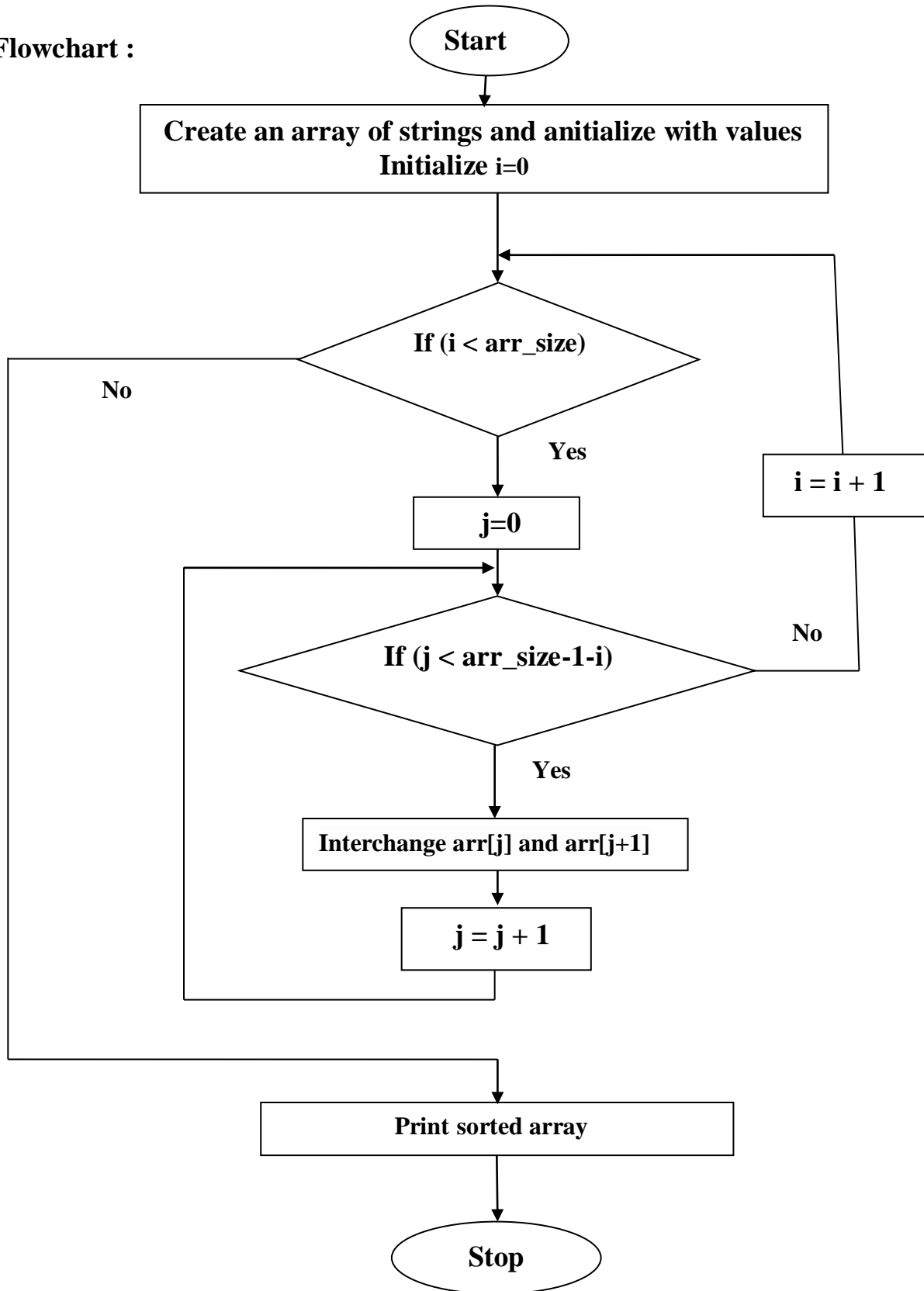
a. If yes, then  $\text{swap}(\text{array}[j], \text{array}[j+1])$

B. End nested loop.

**Step 3:** End outer loop.

**Step 4:** Output the array.

Flowchart :



## VIVA-QUESTIONS:

1. Binary search algorithm cannot be applied to ...

- A. sorted linked list
- B. sorted binary trees
- C. sorted linear array
- D. pointer array

2. Complexity of linear search algorithm is

- A.  $O(n)$
- B.  $O(\log n)$
- C.  $O(n^2)$
- D.  $O(n \log n)$

3. The complexity of bubble sort algorithm is .....

- A.  $O(n)$
- B.  $O(\log n)$
- C.  $O(n^2)$
- D.  $O(n \log n)$