1 CONTENTS

2 QUADRATIC EQUATION (VTU-PROGRAM 1)

2.1 PROBLEM STATEMENT

Design and develop a flowchart or an algorithm that takes three coefficients (a, b, and c) of a Quadratic equation ($ax^2+bx+c=0$) as input and compute all possible roots. Implement a C program for the developed flowchart/algorithm and execute the same to output the possible roots for a given set of coefficients with appropriate messages.

2.2 PLAN OF SOLUTION

A quadratic equation is any equation having the form:

$$ax^2 + bx + c = 0$$

where x represents an unknown, and a, b, and c represent known numbers such that a is not equal to 0. If a = 0, then the equation is linear, not quadratic. The numbers a, b, and c are the coefficients of the equation.

In the quadratic formula, the expression underneath the square root sign is called the discriminant of the quadratic equation.

$$d = b^2 - 4ac$$

The discriminant determines the number and nature of the roots. There are three cases:

- If the discriminant is positive (d > 0), then there are two distinct roots:

$$\frac{-b}{2a} + \frac{\sqrt{d}}{2a} \quad \text{and} \quad \frac{-b}{2a} - \frac{\sqrt{d}}{2a}$$

both of which are real numbers.

- If the discriminant is zero (d = 0), then there is exactly one real root:

$$\frac{-b}{2a} \quad \text{and} \quad \frac{-b}{2a}$$

sometimes called a repeated or double root.

- If the discriminant is negative (d < 0) , then there are no real roots. Rather, there are two distinct (non-real) complex roots:

$$\frac{-b}{2a} + i\ \frac{\sqrt{-d}}{2a} \quad \text{and} \quad \frac{-b}{2a} + i\ \frac{\sqrt{-d}}{2a}$$

which are complex conjugates of each other. In these expressions i is the imaginary unit.

2.3 ALGORITHM

Input: Three non-zero coefficients a, b and c of a Quadratic Equation.

Output: To compute roots for the given Quadratic Equation.

Step 1: Start

Step 2: Read three non-zero coefficients **a, b and c**

Step 3: Check **if a is equal to zero**. If true goto Step 4 otherwise goto Step 5

Step 4: Display the equation is Linear and not Quadratic and goto Step 11.

Step 5: Compute Discriminant:

disc = (b * b) - (4 * a * c)

Step 6: Check **if disc is equal to zero**. If true **goto Step 7** otherwise **goto Step 8**

Step 7: Compute:

root1 = - b / (2.0 * a)

root2 = root1

Display Roots are Real and Equal: root1 and root2 goto Step 11

Step 8: Check **if disc is greater than zero**. If true **goto Step 9** otherwise **goto Step 10**

Step 9: Compute:

root1 = (- b + sqrt (disc)) / (2.0 * a)

root2 = (- b - sqrt (disc)) / (2.0 * a)

Display Roots are Real and Distinct: root1 and root2 goto Step 11

Step 10: Compute:

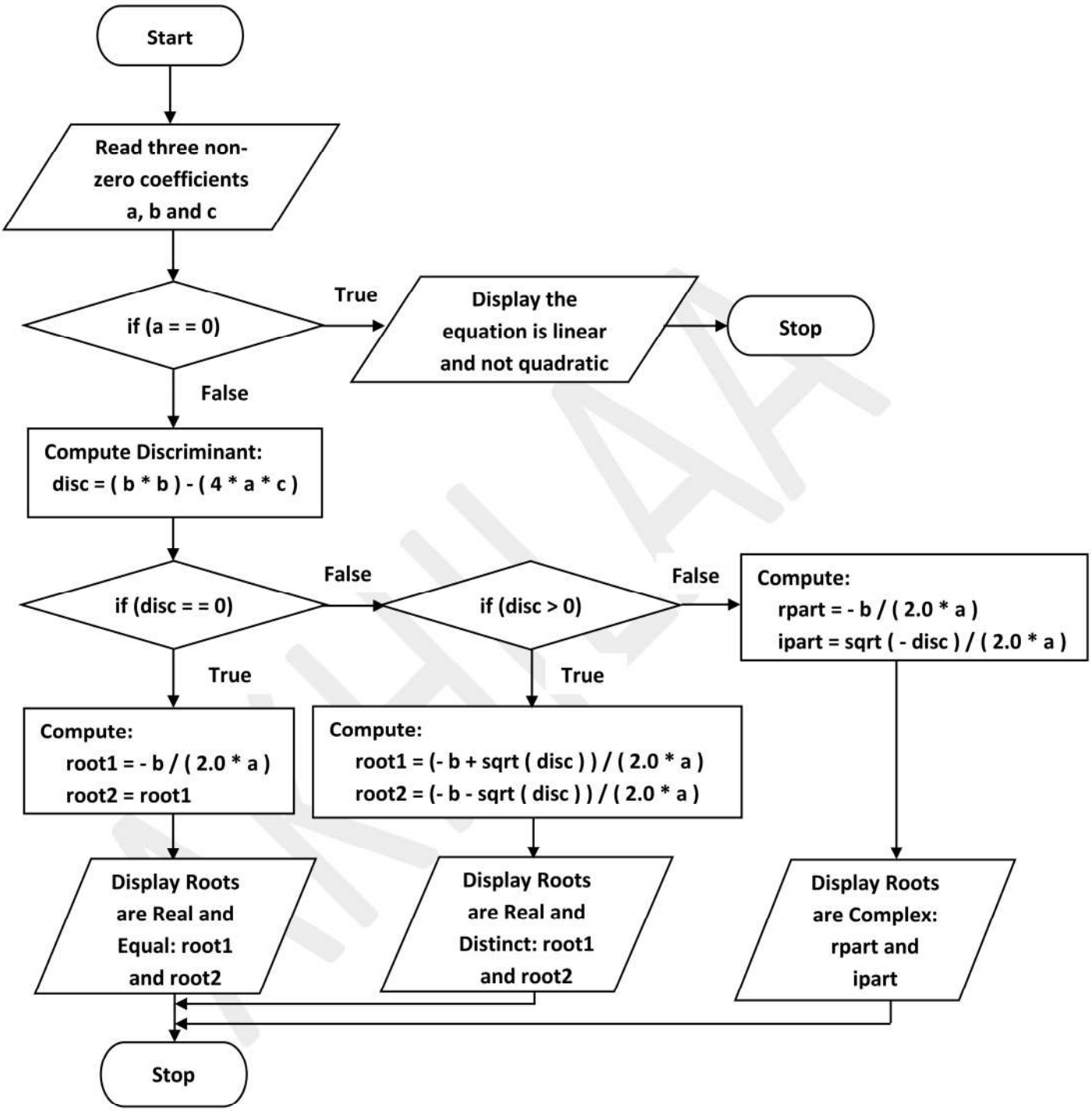
rpart = - b / (2.0 * a)

ipart = sqrt (- disc) / (2.0 * a)

Display Roots are Complex: rpart and ipart goto Step 11

Step 11: Stop

2.4 FLOWCHART



2.5 PROGRAM CODE

```
/*Program to find roots of a Quadratic Equation. */

#include<stdio.h>
#include<math.h>

int main()
{
        float a, b, c, d, rpart, ipart, r1, r2;

        // Input the co-efficients of a, b and c.
        printf ( "\nEnter three non-zero coefficients ( a, b and c ) of the Quadratic
        Equation: " );
        scanf ( "%f%f%f", &a, &b, &c );

        // Check if the equation is quadratic or not.
        if ( a == 0 )
        {
                printf ( "\nThe equation is linear and not quadratic!!!\n" );
                return 1;
        }

        // Compute the discriminant.
        d = ( b * b ) - ( 4 * a * c) ;
        printf ( "\nThe discriminant is: %f\n", d );

        // Compute Real and Equal roots.
        if ( d == 0 )
        {
                r1 = r2 = -b / ( 2.0 * a );
                printf ( "\nRoots are Real and Equal: \n r1= %f \n r2= %f \n\n", r1, r2 );
        }
        // Compute Real and Distinct roots.
        else if ( d > 0 )
        {
                r1 = ( -b - ( sqrt ( d ) ) ) / ( 2.0 * a );
                r2 = ( -b + ( sqrt ( d ) ) ) / ( 2.0 * a );
                printf ( "\nRoots are Real and Distinct: \n r1= %f \n r2= %f \n\n", r1, r2
                );
        }
        // Compute Imaginary roots.
        else
        {
```

```
                rpart = - b / ( 2.0 * a );
                ipart = sqrt ( ( - d ) ) / ( 2.0 * a );
                printf ( "\nRoots are Imaginary: \n r1 = %f + i * %f \n r2 = %f - i * %f
                \n\n", rpart, ipart, rpart, ipart );
        }

        return 0;
}
```

2.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit quadratic_equation.c
akhilaa@akhilaa-VirtualBox:~$ cc quadratic_equation.c -lm
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter three non-zero coefficients ( a, b and c ) of the Quadratic Equation: 1 1 1

The discriminant is: -3.000000

Roots are Imaginary:
 r1 = -0.500000 + i * 0.866025
 r2 = -0.500000 - i * 0.866025

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter three non-zero coefficients ( a, b and c ) of the Quadratic Equation: -1 2 -1

The discriminant is: 0.000000

Roots are Real and Equal:
 r1= 1.000000
 r2= 1.000000

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter three non-zero coefficients ( a, b and c ) of the Quadratic Equation: -1 2 1

The discriminant is: 8.000000

Roots are Real and Distinct:
 r1= 2.414214
 r2= -0.414214

akhilaa@akhilaa-VirtualBox:~$ ./a.out
```

```
Enter three non-zero coefficients ( a, b and c ) of the Quadratic Equation: 1 -1 1

The discriminant is: -3.000000

Roots are Imaginary:
 r1 = 0.500000 + i * 0.866025
 r2 = 0.500000 - i * 0.866025

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter three non-zero coefficients ( a, b and c ) of the Quadratic Equation: 0 5 3

The equation is linear and not quadratic!!!
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter three non-zero coefficients ( a, b and c ) of the Quadratic Equation: 5.1 3.4 9.7

The discriminant is: -186.319992

Roots are Imaginary:
 r1 = -0.333333 + i * 1.338226
 r2 = -0.333333 - i * 1.338226

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter three non-zero coefficients ( a, b and c ) of the Quadratic Equation: 1 6.2 7.4

The discriminant is: 8.839997

Roots are Real and Distinct:
 r1= -4.586607
 r2= -1.613393

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter three non-zero coefficients ( a, b and c ) of the Quadratic Equation: 5 0 3

The discriminant is: -60.000000

Roots are Imaginary:
 r1 = -0.000000 + i * 0.774597
 r2 = -0.000000 - i * 0.774597
```

akhilaa@akhilaa-VirtualBox:~$

2.7 ALTERNATE PROGRAM CODE

/*Program to find roots of a Quadratic Equation. */

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

int main()
{
        float a, b, c, d, rpart, ipart, r1, r2;

        // Input the co-efficients of a, b and c.
        printf ( "\nEnter three non-zero coefficients ( a, b and c ) of the Quadratic Equation: "
        );
        scanf ( "%f%f%f", &a, &b, &c );

        // Check if the equation is quadratic or not.
        if ( a == 0 )
        {
                printf ( "\nThe equation is linear and not quadratic!!!\n" );
                return 1;
        }

        // Compute the discriminant.
        d = ( b * b ) - ( 4 * a * c) ;
        printf ( "\nThe discriminant is: %f\n", d );

        // Compute Real and Equal roots.
        if ( d == 0 )
        {
                r1 = r2 = -b / ( 2.0 * a );
                printf ( "\nRoots are Real and Equal: \n r1= %f \n r2= %f \n\n", r1, r2 );
        }
        // Compute Real and Distinct roots.
        else if ( d > 0 )
        {
                r1 = ( -b - ( sqrt ( d ) ) ) / ( 2.0 * a );
                r2 = ( -b + ( sqrt ( d ) ) ) / ( 2.0 * a );
                printf ( "\nRoots are Real and Distinct: \n r1= %f \n r2= %f \n\n", r1, r2 );

        }
        // Compute Imaginary roots.
```

```
        else
        {
                rpart = - b / ( 2.0 * a );
                ipart = sqrt ( fabs ( d ) ) / ( 2.0 * a );
                printf ( "\nRoots are Imaginary: \n r1 = %f + i * %f \n r2 = %f - i * %f \n\n",
                rpart, ipart, rpart, ipart );
        }

        return 0;
}
```

2.8 VIVA QUESTIONS

- What do you mean by pre-processor directive?
- What is a header file?
- What is main()?
- What are data types?
- What are format specifiers?
- What are printf() and scanf()?
- What are expressions?
- What is if-else ladder and how does it work?
- What is the difference between = and ==?
- What is precedence and associativity?
- What are relational operators?
- What is math header file?
- What is the difference between sqrt(), sqrtf() and sqrtl()?

3 PALINDROME (VTU-PROGRAM 2)

3.1 PROBLEM STATEMENT

Design and develop an algorithm to find the reverse of an integer number NUM and check whether it is PALINDROME or NOT. Implement a C program for the developed algorithm that takes an integer number as input and output the reverse of the same with suitable messages. Ex: Num: 2014, Reverse: 4102, Not a Palindrome.

3.2 PLAN OF SOLUTION

The word palindrome is derived from the Greek 'palin,' or "back" and 'dromos' or "direction". Palindrome is a word, phrase, number, or other sequence of characters which reads the same backward or forward.

For example: 112212211, 1221, 13131, MADAM, MALAYALAM are Palindrome.

Steps to reverse a number:

- Find the remainder by taking the modulus (%) of the number by 10
- Find the reverse by multiplying the previous reversed number with 10 and then add the remainder.
- Divide the number by 10

3.3 ALGORITHM

Input: A positive number.

Output: To check whether the number is PALINDROME or NOT.

Step 1: Start

Step 2: Read an integer **num**

Step 3: Initialize:

temp = num

rev = 0

Step 4: Check **while temp is not equal to zero**. If true **goto Step 5** otherwise **goto Step 6**

Step 5: Compute:

rem = temp % 10

rev = (rev * 10) + rem

temp = temp / 10 **goto Step 4**

Step 6: Display the reversed number

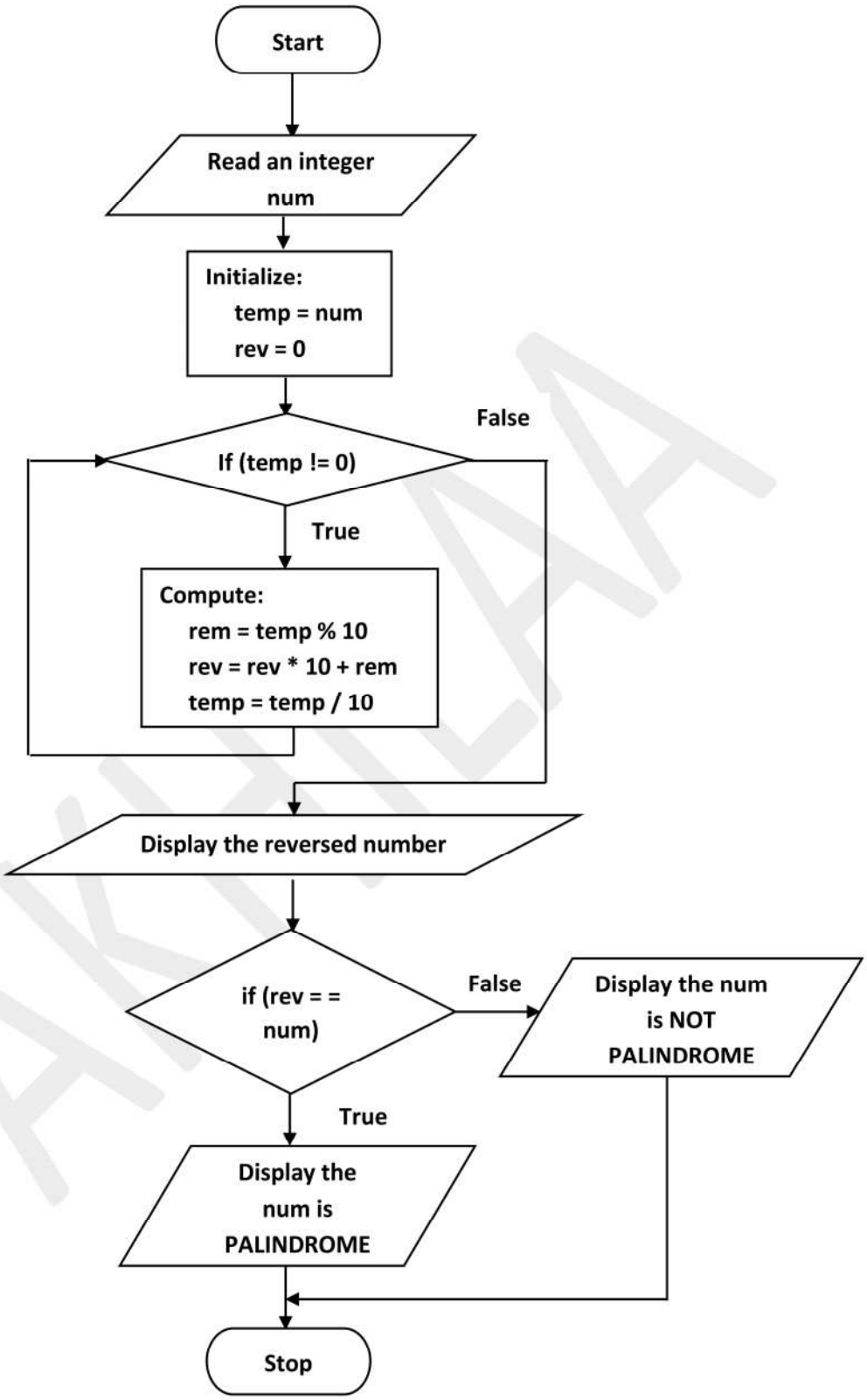
Step 7: Check **if rev is equal to num**. If true **goto Step 8** otherwise **goto Step 9**

Step 8: Display the number is PALINDROME **goto Step 10**

Step 9: Display the number is NOT PALINDROME **goto Step 10**

Step 10: Stop

3.4 FLOWCHART



3.5 PROGRAM CODE

/*Program to check whether the given number is PALINDROME or NOT. */

```
#include<stdio.h>

int main()
{
        int num, temp, rev, rem;

        // Input the number.
        printf ( "\nEnter an integer: " );
        scanf ( "%d", &num );

        // Initialize.
        temp = num;
        rev = 0;

        // Compute reverse of the entered number.
        while ( temp != 0 )
        {
                rem = temp % 10;
                rev = rev * 10 + rem;
                temp = temp / 10;
        }

        // Display reverse number.
        printf ( "\nThe reversed number is: %d\n", rev );

        // Check if the entered number and reversed number are equal.
        if ( rev == num )
                printf ( "\nThe entered number %d is a PALINDROME\n\n", num );
        else
                printf (" \nThe entered number %d is NOT a PALINDROME\n\n", num
                );

        return 0;
}
```

3.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit palindrome.c
akhilaa@akhilaa-VirtualBox:~$ cc palindrome.c
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter an integer: 121121

The reversed number is: 121121

The entered number 121121 is a PALINDROME

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter an integer: 12112

The reversed number is: 21121

The entered number 12112 is NOT a PALINDROME

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter an integer: 33244233

The reversed number is: 33244233

The entered number 33244233 is a PALINDROME

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter an integer: 81899818

The reversed number is: 81899818

The entered number 81899818 is a PALINDROME

akhilaa@akhilaa-VirtualBox:~$
```

3.7 ALTERNATE PROGRAM CODE

/*Program to check whether the given number is PALINDROME or NOT. */

```
#include<stdio.h>

int main()
{

        int num, temp, rev, rem;

        // Input the number.
        printf ( "\nEnter an integer: " );
        scanf ( "%d", &num );

        // Initialize.
        temp = num;
        rev = 0;

        // Compute reverse of the entered number.
        while ( temp > 0 )
        {
                rem = temp % 10;
                rev = rev * 10 + rem;
                temp = temp / 10;
        }

        // Display reverse number.
        printf ( "\nThe reversed number is: %d\n", rev );

        // Check if the entered number and reversed number are equal.
        if ( rev == num )
                printf ( "\nThe entered number %d is a PALINDROME\n\n", num );
        else
                printf (" \nThe entered number %d is NOT a PALINDROME\n\n", num
                );

        return 0;
}
```

3.8 VIVA QUESTIONS

- What happens when a variable is not initialized whenever necessary?
- How does while loop work?
- What is an infinite loop?
- What are arithmetic operators?
- What is the difference between % and /?
- How does if-else statement work?
- Why is the number (in decimal system) 0110 not a palindrome? How should we check these kind of numeral strings for palindrome?

4 SQUARE ROOT (VTU-PROGRAM 3A)

4.1 PROBLEM STATEMENT

Design and develop a flowchart to find the square root of a given number N. Implement a C program for the same and execute for all possible inputs with appropriate messages. Note: Don't use library function sqrt(n).

4.2 PLAN OF SOLUTION

- Suppose we want to find square root of a number, say 's'. Assume that r1 and r2 are roots of x.

  i.e. $\sqrt{x} = \sqrt{r1} * \sqrt{r2}$

  = r1 * r2 where r1 and r2 are equal

  $\sqrt{x}$ = r1 or r2
- Assume an error tolerance of 0.000001
- Assume an initial value of r1. It can be any number other than 1. Here we are assuming r1 = s / 2, because the square root of any number can never be more than half of the number.
- We can obtain r2 in terms of r1 i.e. r2 = s / r1
- When we plot r1 and r2 on the number line, always r1 and r2 lies on the opposite side.
- To bring r1 and r2 closer(equal) to the square root of the given number, we take mean of r1 and r2.

  i.e. r1 = (r1 + r2) /2

  r2 = s / r1
- The above process is repeated until the difference between r1 and r2 becomes very minimal i.e. until the assumed error tolerance is reached.

4.3 ALGORITHM

Input: A positive number.

Output: To compute Square Root of the number.

Step 1: Start

Step 2: Read a number **s** to find the square root

Step 3: Check **if s is less than zero**. If true **goto Step 5** otherwise **goto Step 6**

Step 4: Display Error! Cannot find square root of a negative number and goto Step 9

Step 5: Initialize:

r1 = s / 2.0

r2 = s / r1

diff = fabs (r1 – r2)

Step 6: Compute:

r1 = (r1 + r2) / 2.0

r2 = s / r1

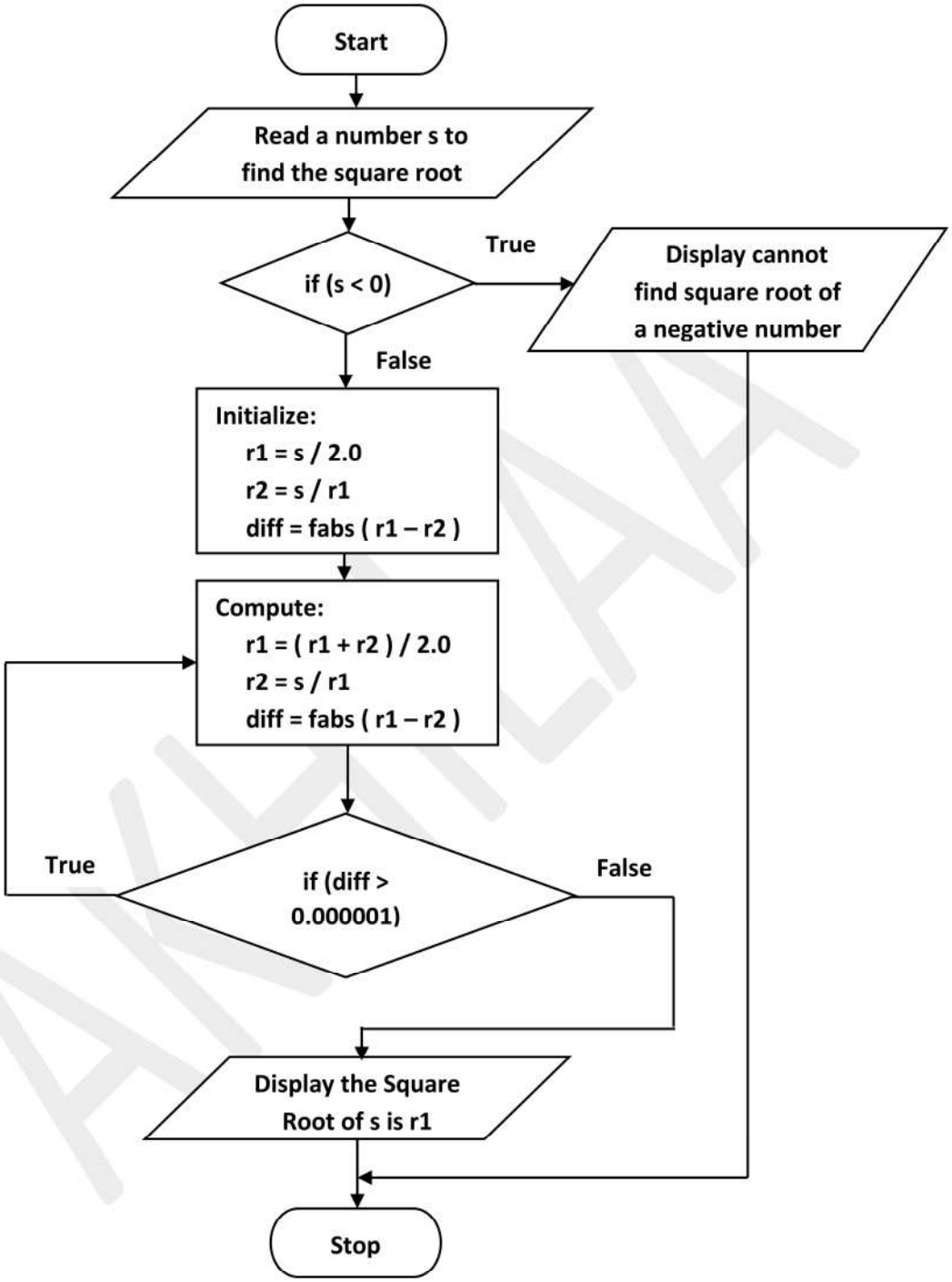
diff = fabs (r1 – r2)

Step 7: Check **while diff is greater than 0.000001**. If true **goto Step 6** otherwise **goto Step 8**

Step 8: Display the Square Root of s is: r1

Step 9: Stop

4.4 FLOWCHART



4.5 PROGRAM CODE

/* Program to find the Square Root of a given number without using built-in function. */

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

int main()
{
        float s, r1, r2, diff;

        // Input the number to find the square root.
        printf ( "\nEnter a number to find the square root of: " );
        scanf ( "%f", &s );

        // Check if the enetered number is negative, since we cannot find the
        square root of a negative number.
        if ( s < 0 )
        {
                printf ( "\nCannot find square root of a negative number!\n\n" );
                return 1;
        }

        // Compute sqrt using iterative method.
        // Guess an initial value for root.
        // Ideally some predifined table lookup should be done to find an optimal
        initial value.

        r1 = s / 2.0;     //Reduces the number of iterations when the number is
                          divided by 2.
        r2 = s / r1;
        diff = fabs ( r1 - r2 );

        // Iterate to compute the root till error is not with-in tolerance.
        do
        {
                // Since on the number line guessed_root and guessed_root_opp
                always lie on the opposites of the actual root,
                // we take the average of the two as the new guessed root.

                r1 = ( r1 + r2 ) / 2.0;

                r2 = s / r1;
```

```
            diff = fabs ( r1 - r2 );

        } while ( diff > 0.000001 );

        // Display the Square Root of the given number.
        printf ( "\nThe Square Root of %f is: %f\n\n", s, r1 );

        return 0;
}
```

4.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit square_root.c
akhilaa@akhilaa-VirtualBox:~$ cc square_root.c
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter a number to find the square root of: 25

The Square Root of 25.000000 is: 5.000000

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter a number to find the square root of: -49

Cannot find square root of a negative number!

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter a number to find the square root of: 10

The Square Root of 10.000000 is: 3.162278

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter a number to find the square root of: 625

The Square Root of 625.000000 is: 25.000000

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter a number to find the square root of: 33

The Square Root of 33.000000 is: 5.744563
```

akhilaa@akhilaa-VirtualBox:~$

4.7 ALTERNATE PROGRAM CODE

/* Program to find the Square Root of a given number without using built-in function. */

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

int main()
{
        float s, r1, r2, diff;

        // Input the number to find the square root.
        printf ( "\nEnter a number to find the square root of: " );
        scanf ( "%f", &s );

        // Check if the enetered number is negative, since we cannot find the
        square root of a negative number.
        if ( s < 0 )
        {
                printf ( "\nCannot find square root of a negative number!\n\n" );
                return 1;
        }

        // Compute sqrt using iterative method.
        // Guess an initial value for root.
        // Ideally some predifined table lookup should be done to find an optimal
        initial value.

        r1 = s / 2.0;     //Reduces the number of iterations when the number is
                          divided by 2.
        r2 = s / r1;
        diff = fabs ( r1 - r2 );

        // Iterate to compute the root till error is not with-in tolerance.
        while ( diff > 0.000001 )
        {
                // Since on the number line guessed_root and guessed_root_opp
                always lie on the opposites of the actual root,
                // we take the average of the two as the new guessed root.
```

```
            r1 = ( r1 + r2 ) / 2.0;

            r2 = s / r1;

            diff = fabs ( r1 - r2 );

        }

        // Display the Square Root of the given number.
        printf ( "\nThe Square Root of %f is: %f\n\n", s, r1 );

        return 0;
}
```

4.8 VIVA QUESTIONS

- What is #define?
- What are fabsf(), abs(), fabs(), fabsl()?
- Why should we not use abs() with floating point variables?
- How should we compare two floating point numbers for equality?
- What do you mean by error tolerance or round off error?
- How does do-while loop works?
- What is the difference between while and do-while loop?

5 LEAP YEAR (VTU-PROGRAM 3B)

5.1 PROBLEM STATEMENT

Design and develop a C program to read a year as an input and find whether it is leap year or not. Also consider end of the centuries.

5.2 PLAN OF SOLUTION

A leap year is a year containing one additional day in order to keep the calendar year synchronized with the astronomical or seasonal year. Because seasons and astronomical events do not repeat in a whole number of days, calendars that have the same number of days in each year, over time, drift with respect to the event that the year is supposed to track. By inserting an additional day or month into the year, the drift can be corrected. A year that is not a leap year is called a common year.

- Rule 1: If year is divisible by 400.

  For example: 1600, 2000 etc are leap year while 1500, 1700 are not leap year.

- Rule 2: If year is not divisible by 100 but it is divisible by 4 then that year is a leap year.

  For example: 2004, 2008, 1012 are leap year.

5.3 ALGORITHM

Input: An year.

Output: To check whether the year is LEAP YEAR or NOT.

Step 1: Start

Step 2: Read an **year**

Step 3: Check **if year is divisible by 4 and year is not divisible by 100**. If true **goto Step 4** otherwise goto **Step 5**

Step 4: Display the year is LEAP YEAR **goto Step 8**

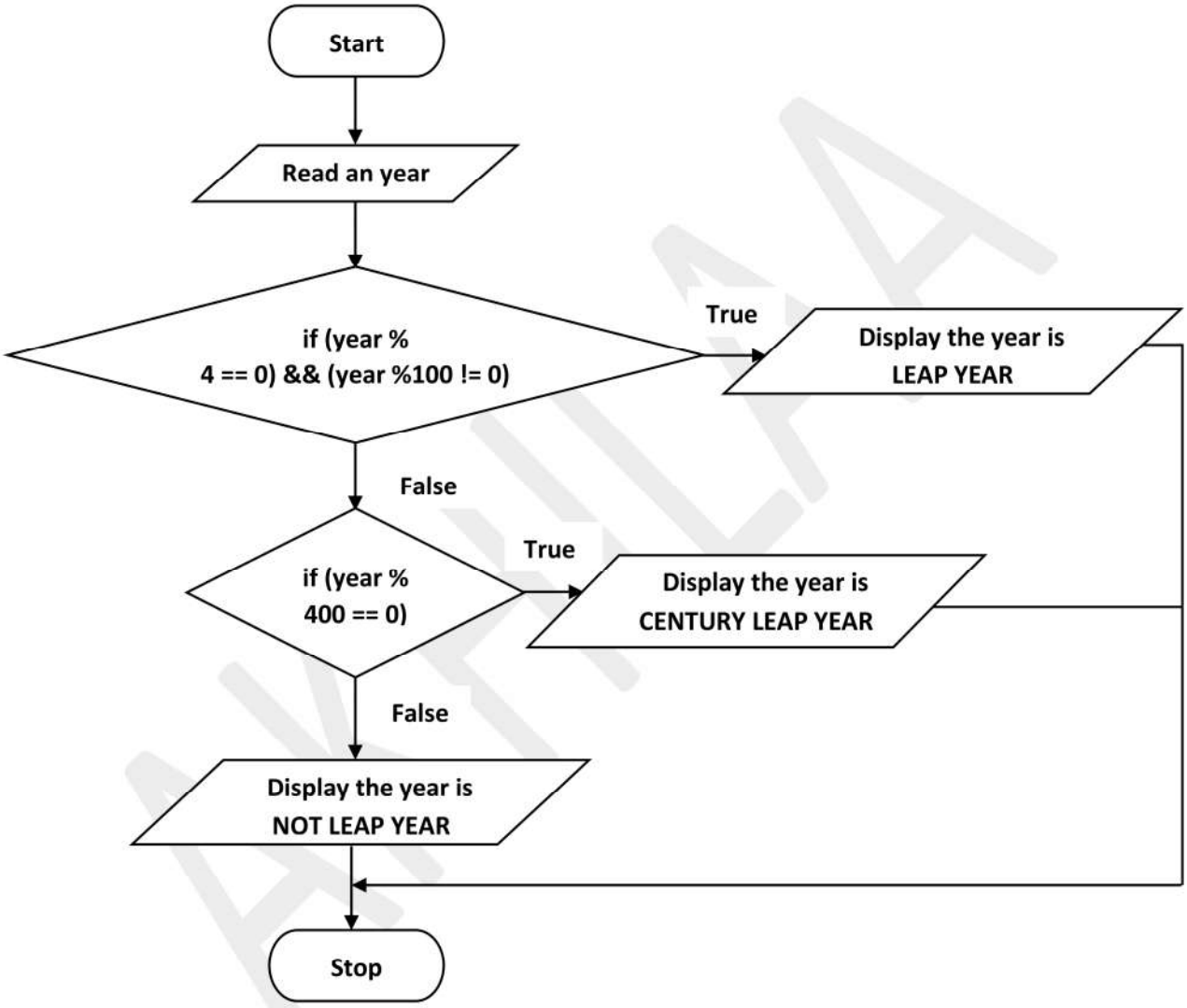
Step 5: Check **if year is divisible by 400**. If true **goto Step 6** otherwise **goto Step 7**

Step 6: Display the year is CENTURY LEAP YEAR **goto Step 8**

Step 7: Display the year is NOT LEAP YEAR **goto Step 8**

Step 8: Stop

5.4 FLOWCHART



5.5 PROGRAM CODE

/* Program to check whether a given year is LEAP YEAR or NOT. */

```
#include <stdio.h>

int main()
{
        int year;
```

```
        // Input an year.
        printf ( "\nEnter an year: " );
        scanf ( "%d", &year );

        if ( ( year % 4 == 0 ) && ( year % 100 != 0 ) )
        {
                printf ( "\n%d year is a LEAP YEAR\n\n", year );
        }
        else if ( year % 400 == 0 )
        {
                printf ( "\n%d year is a CENTURY LEAP YEAR\n\n", year );
        }
        else
        {
                printf ( "\n%d year is NOT a LEAP YEAR\n\n", year );
        }

        return 0;
}
```

5.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit leap_year.c
akhilaa@akhilaa-VirtualBox:~$ cc leap_year.c
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter an year: 1500

1500 year is NOT a LEAP YEAR

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter an year: 1600

1600 year is a CENTURY LEAP YEAR

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter an year: 1700

1700 year is NOT a LEAP YEAR

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter an year: 2000
```

2000 year is a CENTURY LEAP YEAR

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter an year: 2011

2011 year is NOT a LEAP YEAR

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter an year: 2016

2016 year is a LEAP YEAR

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter an year: 2020

2020 year is a LEAP YEAR

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter an year: 2025

2025 year is NOT a LEAP YEAR

akhilaa@akhilaa-VirtualBox:~$

5.7 ALTERNATE PROGRAM CODES

- **/* Program to check whether a given year is LEAP YEAR or NOT. */**

```
#include <stdio.h>

int main()
{
        int year;

        // Input an year.
        printf ( "\nEnter an year: " );
        scanf ( "%d", &year );

        if ( ( ( year % 4 == 0 ) && ( year % 100 != 0 ) ) || ( year % 400 == 0 ) )
        {
                printf ( "\n%d year is a LEAP YEAR\n\n", year );
        }
        else
```

```
        {
                printf ( "\n%d year is NOT a LEAP YEAR\n\n", year );
        }

        return 0;
}
```

- **/* Program to check whether a given year is LEAP YEAR or NOT. */**

```
#include <stdio.h>

int main()
{
        int year;

        // Input an year.
        printf ( "\nEnter an year: " );
        scanf ( "%d", &year );

        if ( year % 4 != 0 )
        {
                printf ( "\n%d is NOT a LEAP YEAR\n\n", year );
        }
        else
        {
                // Checking for a century year.
                if ( year % 100 != 0 )
                {
                        printf ( "\n%d is a LEAP YEAR\n\n", year );
                }
                else
                {
                        if ( year % 400 != 0 )
                                printf ( "\n%d is NOT a LEAP YEAR\n\n", year );
                        else
                                printf ( "\n%d is a LEAP YEAR\n\n", year );
                }
        }

        return 0;
}
```

5.8 VIVA QUESTIONS

- How does nested if-else work?
- What are logical operators?

6 POLYNOMIAL EVALUATION (VTU-PROGRAM 4)

6.1 PROBLEM STATEMENT

Design and develop an algorithm to evaluate polynomial $f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$, for a given value of x and its coefficients using Horner's method. Implement a C program for the same and execute the program with different set of values of coefficients and x.

6.2 PLAN OF SOLUTION

A polynomial is expressed as:

$$f(x) = \sum_{k=0}^{n} a^k x^k$$

or

$$f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_kx^k$$

where a_k are real numbers representing the polynomial coefficients and x^k are the polynomial variables.

The above polynomial is said to be of the n^{th} degree, i.e. *deg (f (x)) = n* where *n* represents the highest variable exponent.

Horner's rule for polynomial division is an algorithm used to simplify the process of evaluating a polynomial *f(x)* at a certain value $x = x_0$ by dividing the polynomial into monomials (polynomials of the 1^{st} degree). Each monomial involves a maximum of one multiplication and one addition processes. The result obtained from one monomial is added to the result obtained from the next monomial and so forth in an accumulative addition fashion. This division process is also known as synthetic division.

To explain the above, re-write the polynomial in its expanded form;

$$f(x_0) = a_0 + a_1x_0 + a_2x_0^2 + \ldots + a_nx_0^n$$

This can, also, be written as:

$$f(x_0) = a_0 + x_0(a_1 + x_0(a_2 + x_0(a_3 + ... + (a_{n-1} + a_n x_0)....)$$

For example:

Evaluate: $f(x) = 2x^3 - 6x^2 + 2x - 1$ for $x = 3$

Using Synthetic Division:

x |	x^3	x^2	x^1	x^0
3 |	2	-6	2	-1
|	+	+	+	+
|	0	6	0	6
		(3 * 2)	(3 * 0)	(3 * 2)
---	---	---	---	---
	2	0	2	**5** → Answer

6.3 ALGORITHM

Input: A Polynomial Equation.

Output: To compute the Sum of Polynomial using Horner's method.

Step 1: **Start**

Step 2: Read the **order** of polynomial

Step 3: Read the **coefficients (order + 1)** of polynomial starting with the lowest order coefficient first

Step 4: Read the value of **x**

Step 5: Check **if order is equal to zero.** If true **goto step 6** otherwise **goto Step 7**

Step 6: **Display the sum of polynomial is a[0]** **goto Step 12**

Step 7: Initialize:

sum = coefficients [order] * x

i = order - 1

Step 8: Check **if i is greater than or equal to zero**. If true **goto Step 10** otherwise **goto**

Step 11

Step 9: Compute:

sum = (sum + coefficients [i]) * x

Decrement i **goto Step 9**

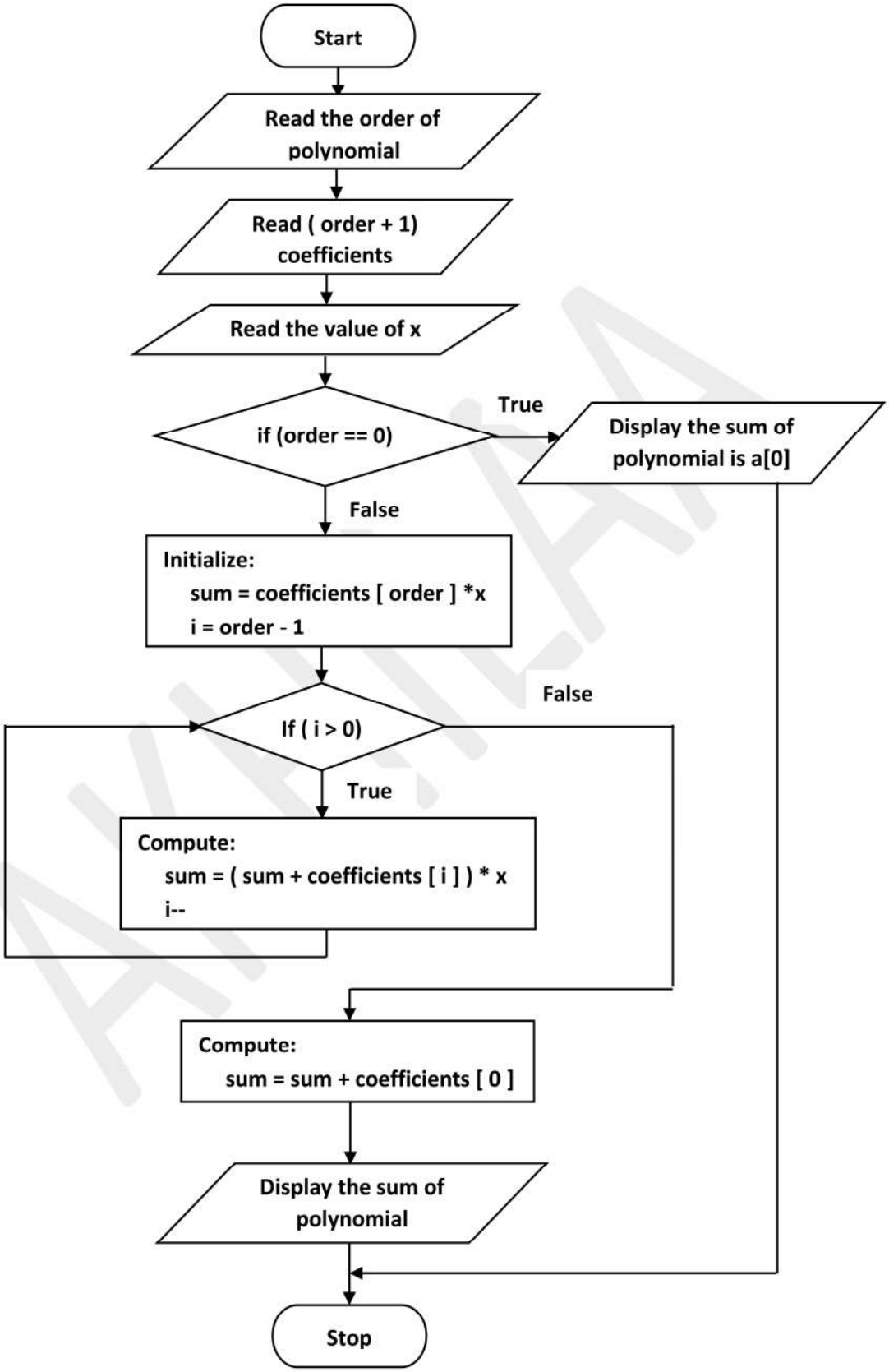
Step 10: Compute:

sum = sum + coefficients [0]

Step 11: Display the sum of polynomial

Step 12: Stop

6.4 FLOWCHART



6.5 PROGRAM CODE

```
/* Program to evaluate a polynomial using Horner's method. */

#include<stdio.h>

int main()
{
        int order, i;
        float a [ 15 ], x, sum;

        // Input the order of polynomial.
        printf ( "\nEnter the order of polynomial: " );
        scanf ( "%d", &order );

        // Input the coefficients starting from lowest order.
        printf ( "\nEnter %d co-efficients of polynomial, starting with lowest order
        coefficient first:\n", (order+1) );
        for ( i = 0 ; i <= order ; i++ )
        {
                scanf ( "%f", &a [ i ] );
        }

        // Input the value of x.
        printf ( "\nEnter the value of x: " );
        scanf ( "%f", &x );

        // Check if the order of the polynomial is zero.
        if ( order == 0 )
        {
                printf ( "\nThe sum of polynomial f(%f): %f\n\n", x, a [ 0 ] );
                return 0;
        }

        // Initialize sum to the highest order coefficient.
        sum = a [ order ] * x;

        // Compute sum using Horner's method.
        for ( i = order - 1 ; i > 0 ; i-- )
        {
                sum = ( sum + a [ i ] ) * x;
        }
        // Add the constant a (a0) to the sum.
        sum = sum + a [ 0 ];

        // Display the sum of the given polynomial.
        printf ( "\nThe sum of polynomial f(%f): %f\n\n", x, sum );
```

```
        return 0;
}
```

6.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit polynomial_evaluation.c
akhilaa@akhilaa-VirtualBox:~$ cc polynomial_evaluation.c
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the order of polynomial: 0

Enter 1 co-efficients of polynomial, starting with lowest order coefficient first:
5

Enter the value of x: 2

The sum of polynomial f(2.000000): 5.000000

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the order of polynomial: 4

Enter 5 co-efficients of polynomial, starting with lowest order coefficient first:
1
2
3
4
5

Enter the value of x: 2

The sum of polynomial f(2.000000): 129.000000

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the order of polynomial: 4

Enter 5 co-efficients of polynomial, starting with lowest order coefficient first:
5
4
3
2
1

Enter the value of x: 2
```

```
The sum of polynomial f(2.000000): 57.000000

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the order of polynomial: 3

Enter 4 co-efficients of polynomial, starting with lowest order coefficient first:
1.0
2.1
3.2
4.3

Enter the value of x: 3.3

The sum of polynomial f(3.300000): 197.307098

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the order of polynomial: 2

Enter 3 co-efficients of polynomial, starting with lowest order coefficient first:
1.8
9.7
2.9

Enter the value of x: 8.1

The sum of polynomial f(8.100000): 270.639008

akhilaa@akhilaa-VirtualBox:~$
```

6.7 ALTERNATE PROGRAM CODE

```
/* Program to evaluate a polynomial using Horner's method. */

#include<stdio.h>

int main()
{
        int order, i;
        float a [ 15 ], x, sum;

        // Input the order of polynomial.
        printf ( "\nEnter the order of polynomial: " );
        scanf ( "%d", &order );

        // Input the coefficients starting from lowest order.
```

```
        printf ( "\nEnter %d co-efficients of polynomial, starting with lowest order
        coefficient first:\n", (order+1) );
        for ( i = 0 ; i <= order ; i++ )
        {
                scanf ( "%f", &a [ i ] );
        }

        // Input the value of x.
        printf ( "\nEnter the value of x: " );
        scanf ( "%f", &x );

        // Check if the order of the polynomial is zero.
        if ( order == 0 )
        {
                printf ( "\nThe value of polynomial f(%f): %f\n\n", x, a [ 0 ] );
                return 0;
        }

        // Add the constant a (a0) to the sum.
        sum = a [ order ];

        // Compute sum using Horner's method.
        for ( i = order-1 ; i >= 0 ; i-- )
        {
                sum = ( sum * x ) + a [ i ] ;
        }

        // Display the sum of the given polynomial.
        printf ( "\nThe value of polynomial f(%f): %f\n\n", x, sum );

        return 0;
}
```

6.8 VIVA QUESTIONS

- How does for loop works?
- What do you mean by post-increment and pre-increment operators?
- What is an array?
- How to declare an array?
- What is the size of an array?
- What are post-decrement and pre-decrement operators?

7 TAYLOR SERIES (VTU-PROGRAM 5)

7.1 PROBLEM STATEMENT

Draw the flowchart and Write a C Program to compute Sin(x) using Taylor series approximation given by Sin(x) = x - (x^3/3!) + (x^5/5!) - (x^7/7!) + ... Compare your result with the built- in Library function. Print both the results with appropriate messages.

7.2 PLAN OF SOLUTION

- Assume an error tolerance of 0.000001f.
- Compute the value of x in radians given the degree.

  x = (π / 180) * degree
- Since the first term in the Taylor Series for Sin(x) is x, initialize the term and sum to x.
- The series is in terms of odd powers of x. So to find the second term in the series (x^3/3!), initialize power which is represented as i to 3.
- Compute the second term (-x^3/3!) by appending a negative sign and multiplying the first term (x) with x two times and dividing it by i.
- Initialize last_sum to the previous sum value and compute the new sum by adding the previous sum value and current term value.
- Increment the value of i by 2 since the series is in terms of odd powers of x.
- Repeat the above process until the absolute value of last_sum and sum value is greater than the assumed error tolerance.

7.3 ALGORITHM

Input: The value of x in degree.

Output: To compute Sin (x) using Taylor Series.

Step 1: Start

Step 2: Read the value of **x in degree**

Step 3: Compute the value **x** in **radians**:

x = (3.1412 / 180.0) * degree

Step 4: Initialize:

term = x

sum = term

i = 3

Step 5: Compute:

term = (- term * x * x) / (i * (i - 1))

last_sum = sum

sum = sum + term

i = i + 2

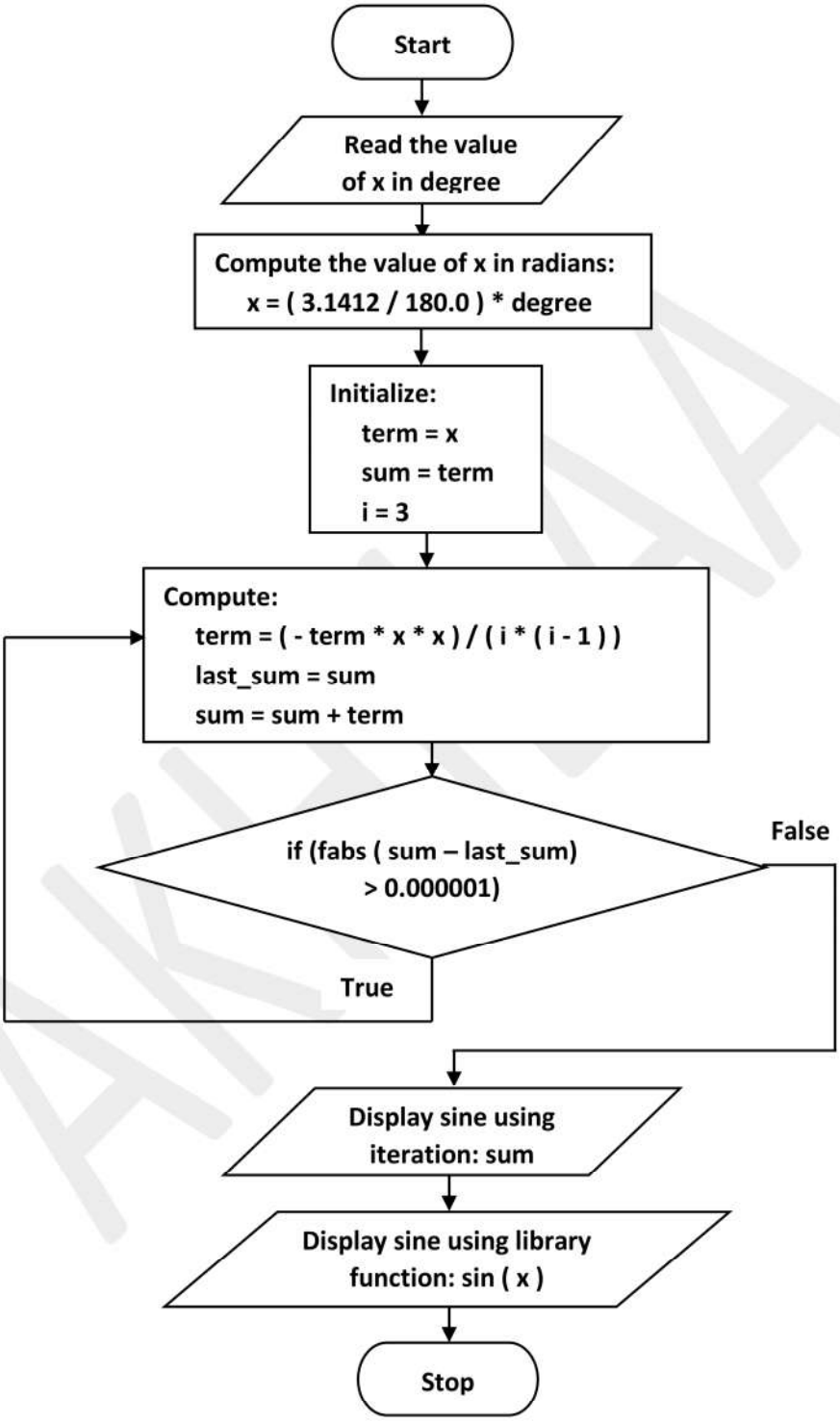
Step 6: Check **while the absolute value of difference of sum and last_sum is greater than 0.000001**. If true **goto Step 5** otherwise **goto Step 7**

Step 7: Display sine using iteration: sum

Step 8: Display sine using library function: sin (x)

Step 9: Stop

7.4 FLOWCHART



7.5 PROGRAM CODE

```
/* Program to compute Sin(x) using Taylor Series. */

#include<stdio.h>
#include<math.h>

int main()
{
        int i;
        float degree, x, term, sum, last_sum;

        // Input the value of x in degree.
        printf ( "\nEnter the value of x in degree: " );
        scanf ( "%f", &degree );

        // Compute the value of x in radians.
        x = (3.1412/180.0) * degree;

        // Initialize the value of term and sum to first term i.e x in the series and
        // since the degree of next term is starting from 3, the value of i is 3.
        term = x;
        sum = term;
        i = 3;

        // Iterate to compute the sum of taylor series till error is not with-in
        tolerance.
        do
        {
                term = ( - term * x * x ) / ( i * ( i - 1 ) );
                last_sum = sum;
                sum = sum + term;
                i = i + 2;          // Since the degree of terms is odd.
        } while ( fabs ( sum - last_sum ) > 0.00001 );

        // Display the value of sine using iteration.
        printf ( "\nsin using iteration      : %f\n", sum );

        // Compare the value of sine computed using iteration with in-built library
        function.
        printf ( "\nsin using library function: %f\n\n", sinf ( x ) );

        return 0;
}
```

7.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit taylor_series.c
akhilaa@akhilaa-VirtualBox:~$ cc taylor_series.c -lm
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the value of x in degree: 30

sin using iteration      : 0.499943

sin using library function: 0.499943

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the value of x in degree: 45

sin using iteration      : 0.707037

sin using library function: 0.707037

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the value of x in degree: 60

sin using iteration      : 0.865960

sin using library function: 0.865960

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the value of x in degree: 90

sin using iteration      : 1.000000

sin using library function: 1.000000

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the value of x in degree: 55.5

sin using iteration      : 0.824058

sin using library function: 0.824058

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the value of x in degree: 72.3
```

```
sin using iteration      : 0.952614

sin using library function: 0.952614

akhilaa@akhilaa-VirtualBox:~$
```

7.7 ALTERNATE PROGRAM CODE

```
/* Program to compute Sin(x) using Taylor Series. */

#include<stdio.h>
#include<math.h>

int main()
{
        int i;
        float degree, x, term, sum, last_sum;

        // Input the value of x in degree.
        printf ( "\nEnter the value of x in degree: " );
        scanf ( "%f", &degree );

        // Compute the value of x in radians.
        x = (3.1412/180.0) * degree;

        // Initialize the value of term and sum to first term i.e x in the series and
        // since the degree of next term is starting from 3, the value of i is 3.
        term = x;
        sum = term;

        // Iterate to compute the sum of taylor series till error is not with-in
        tolerance.
        for ( i = 3; fabs ( term ) > 0.000001 ; i = i + 2 )
        {
                term = ( - term * x * x ) / ( i * ( i - 1 ) );
                sum = sum + term;
        }
        // Display the value of sine using iteration.
        printf ( "\nsin using iteration      : %f\n", sum );

        // Compare the value of sine computed using iteration with in-built library
        function.
        printf ( "\nsin using library function: %f\n\n", sinf ( x ) );

        return 0;
}
```

7.8 VIVA QUESTIONS

- What is the difference between sin(), sinf() and sinl()?

8 BUBBLE SORT (VTU-PROGRAM 6)

8.1 PROBLEM STATEMENT

Develop an algorithm, implement and execute a C program that reads N integer numbers and arrange them in ascending order using Bubble Sort.

8.2 PLAN OF SOLUTION

Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm, which is a comparison sort, is named for the way smaller elements "bubble" to the top of the list.

"5 1 4 2 8", and sort the array from lowest number to greatest number using bubble sort. In each step, elements written in bold are being compared. Three passes will be required.

FIRST PASS:

(**5 1** 4 2 8) → (**1 5** 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1.

(1 **5 4** 2 8) → (1 **4 5** 2 8), Swap since 5 > 4

(1 4 **5 2** 8) → (1 4 **2 5** 8), Swap since 5 > 2

(1 4 2 **5 8**) → (1 4 2 **5 8**), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

SECOND PASS:

(**1 4** 2 5 8) → (**1 4** 2 5 8)

(1 **4 2** 5 8) → (1 **2 4** 5 8), Swap since 4 > 2

(1 2 **4 5** 8) → (1 2 **4 5** 8)

(1 2 4 **5 8**) → (1 2 4 **5 8**)

Now, the array is already sorted, but the algorithm does not know if it is

completed. The algorithm needs one **whole** pass without **any** swap to know it is sorted.

THIRD PASS:

(**1 2** 4 5 8) → (**1 2** 4 5 8)

(1 **2 4** 5 8) → (1 **2 4** 5 8)

(1 2 **4 5** 8) → (1 2 **4 5** 8)

(1 2 4 **5 8**) → (1 2 4 **5 8**)

8.3 ALGORITHM

Input: An array of integers entered in random order.

Output: To sort the array in ascending order using Bubble Sort.

Step 1: Start

Step 2: Read the number of integers: **n**

Step 3: Read the numbers: **a [n]**

Step 4: Initialize:

i = 0

j = 0

Step 5: Check **if i is less than n - 1.** If true **goto Step 6** otherwise **goto Step 11**

Step 6: Check **if j is less than n - 1 - i**. If true **goto Step 7** otherwise **goto Step 10**

Step 7: Check **if a [j] is greater than a [j + 1]**. If true **goto Step 8** otherwise **goto Step 9**

Step 8: Compute:

temp = a [j]

a [j] = a [j + 1]

a [j + 1] = temp

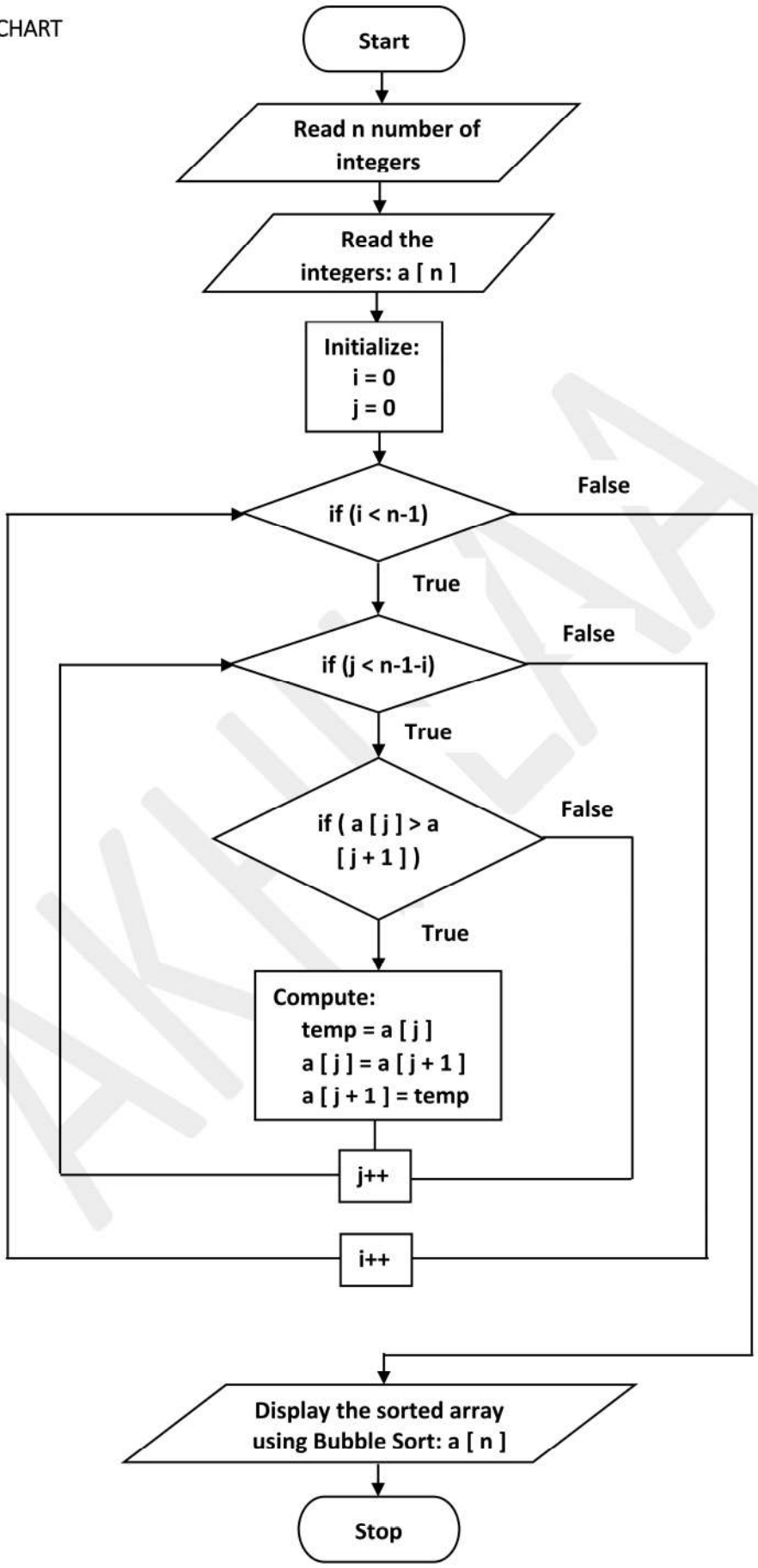
Step 9: Increment j **goto Step 6**

Step 10: Increment i **goto Step 5**

Step 11: Display the sorted array using Bubble Sort: a [n]

Step 12: Stop

8.4 FLOWCHART



8.5 PROGRAM CODE

```
/* Program to sort numbers using Bubble Sort. */

#include<stdio.h>

int main()
{
        int a [ 25 ], i, j, n, temp;

        // Input the number of integers that you want to sort.
        printf ( "\nEnter the number of integers to be sorted: " );
        scanf ( "%d" , &n);

        // Input the numbers.
        printf ( "\nEnter the numbers:\n" );
        for ( i = 0 ; i < n ; i++ )
        {
                scanf ( "%d" , &a [ i ] );
        }

        // Bubble Sort.
        for ( i = 0 ; i < n - 1 ; i++ )        // Passes.
        {
                for ( j = 0 ; j < n - 1 - i ; j++ )              // Comparisions.
                {
                        if ( a [ j ] > a [ j + 1 ] )   // If jth element is greater than j+1th
                                                       element then swap.
                        {
                                temp = a [ j ];
                                a [ j ] = a [ j + 1 ];
                                a [ j + 1 ] = temp;
                        }
                }
        }

        // Display sorted array in ascending order.
        printf ( "\n\nThe sorted array using Bubble Sort is:\n" );
        for ( i = 0 ; i < n ; i++ )
        {
                printf ( "%d\t", a [ i ] );
        }
        printf ( "\n\n" );
```

```
        return 0;
}
```

8.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit bubble_sort.c
akhilaa@akhilaa-VirtualBox:~$ cc bubble_sort.c
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number of integers to be sorted: 5

Enter the numbers:
5
1
3
4
2

The sorted array using Bubble Sort is:
1 2       3       4       5

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number of integers to be sorted: 6

Enter the numbers:
23
55
-10
19
6
32

The sorted array using Bubble Sort is:
-10       6       19      23      32      55

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number of integers to be sorted: 7

Enter the numbers:
5
3
8
```

```
1
9
2
6

The sorted array using Bubble Sort is:
1 2       3       5       6       8       9

akhilaa@akhilaa-VirtualBox:~$
```

8.7 ALTERNATE PROGRAM CODE

➢ **/* Program to sort numbers using Bubble Sort. */**

```
#include<stdio.h>

int main()
{
        int a [ 25 ], i, j, n, temp;

        // Input the number of integers that you want to sort.
        printf ( "\nEnter the number of integers to be sorted: " );
        scanf ( "%d" , &n);

        // Input the numbers.
        printf ( "\nEnter the numbers:\n" );
        for ( i = 0 ; i < n ; i++ )
        {
                scanf ( "%d" , &a [ i ] );
        }

        // Bubble Sort.
        for ( i = 1 ; i < n ; i++ )  // Passes.
        {
                for ( j = 0 ; j < n - i ; j++ )                 // Comparisions.
                {
                        if ( a [ j ] > a [ j + 1 ] )   // If jth element is greater than j+1th
                                                       element then swap.
                        {
                                temp = a [ j ];
                                a [ j ] = a [ j + 1 ];
                                a [ j + 1 ] = temp;
                        }
                }
```

```
        }

        // Display sorted array in ascending order.
        printf ( "\n\nThe sorted array using Bubble Sort is:\n" );
        for ( i = 0 ; i < n ; i++ )
        {
                printf ( "%d\t", a [ i ] );
        }
        printf ( "\n\n" );

        return 0;
}
```

➢ **/* Program to sort numbers using Bubble Sort. */**

```
#include<stdio.h>

int main()
{
        int a [ 25 ], i, j, k, n, temp;

        // Input the number of integers that you want to sort.
        printf ( "\nEnter the number of integers to be sorted: " );
        scanf ( "%d" , &n);

        // Input the numbers.
        printf ( "\nEnter the numbers:\n" );
        for ( i = 0 ; i < n ; i++ )
        {
                scanf ( "%d" , &a [ i ] );
        }


        // Bubble Sort.
        for ( i = 0 ; i < n - 1 ; i++ )// Passes.
        {
                for ( j = 0 ; j < n - i - 1 ; j++ )          // Comparisions.
                {
                        if ( a [ j ] > a [ j + 1 ] )    // If jth element is greater than j+1th
                                                        element then swap.
                        {
                                temp = a [ j ];
                                a [ j ] = a [ j + 1 ];
                                a [ j + 1 ] = temp;
                        }
                }
                printf ( "\nAfter Pass %d: ", i );
```

```
                for ( k = 0 ; k < n ; k ++ )
                {
                        printf ( "\t%d\t" , a [ k ] );
                }
        }

        // Display sorted array in ascending order.
        printf ( "\n\nThe sorted array using Bubble Sort is:\n" );
        for ( i = 0 ; i < n ; i++ )
        {
                printf ( "%d\t", a [ i ] );
        }
        printf ( "\n\n" );

        return 0;

}
```

Output:

```
akhilaa@akhilaa-VirtualBox:~$ gedit bubble_sort.c
akhilaa@akhilaa-VirtualBox:~$ cc bubble_sort.c
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number of integers to be sorted: 5

Enter the numbers:
5
1
3
4
2

After Pass 0:   1       3       4       2       5
After Pass 1:   1       3       2       4       5
After Pass 2:   1       2       3       4       5
After Pass 3:   1       2       3       4       5

The sorted array using Bubble Sort is:
1 2     3       4       5

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number of integers to be sorted: 6

Enter the numbers:
23
```

```
55
-10
19
6
32

After Pass 0:   23          -10         19          6           32
  55
After Pass 1:   -10         19          6           23          32
  55
After Pass 2:   -10         6           19          23          32
  55
After Pass 3:   -10         6           19          23          32
  55
After Pass 4:   -10         6           19          23          32
  55

The sorted array using Bubble Sort is:
-10     6       19      23      32      55

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number of integers to be sorted: 7

Enter the numbers:
5
3
8
1
9
2
6

After Pass 0:   3           5           1           8           2
  6           9
After Pass 1:   3           1           5           2           6
  8           9
After Pass 2:   1           3           2           5           6
  8           9
After Pass 3:   1           2           3           5           6
  8           9
After Pass 4:   1           2           3           5           6
  8           9
```

After Pass 5: 1 2 3 5 6
8 9

The sorted array using Bubble Sort is:
1 2 3 5 6 8 9

akhilaa@akhilaa-VirtualBox:~$

8.8 VIVA QUESTIONS

- How does nested for loop works?

9 MATRIX MULTIPLICATION (VTU-PROGRAM 7)

9.1 PROBLEM STATEMENT

Develop, implement and execute a C program that reads two matrices A (m x n) and B (p x q) and Compute product of matrices A and B. Read matrix A and matrix B in row major order and in column major order respectively. Print both the input matrices and resultant matrix with suitable headings and output should be in matrix format only. Program must check the compatibility of orders of the matrices for multiplication. Report appropriate message in case of incompatibility.

9.2 PLAN OF SOLUTION

- Consider two matrices A of order m x n (m number of rows and n number of columns) and B of order p x q (p number of rows and q number of columns).
- Matrix multiplication is possible only when the number of columns in first matrix has to be equal to number of rows is second matrix (n==p). i.e., (m x n) (n x q) = (m x q).
- The product C (m X q)of two matrices A and B is defined as: $C_{ik} = A_{ij} B_{jk}$

$$\begin{bmatrix} C_{11} & C_{12} & C_{1q} \\ C_{21} & C_{22} & C_{2q} \\ C_{m1} & C_{m2} & C_{mq} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{1n} \\ A_{21} & A_{22} & A_{2n} \\ A_{m1} & A_{m2} & A_{mn} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} & B_{1q} \\ B_{21} & B_{22} & B_{2q} \\ B_{n1} & B_{n2} & B_{nq} \end{bmatrix}$$

where,

$C_{11} = A_{11} B_{11} + A_{12} B_{21} + \ldots + A_{1n} B_{n1}$

$C_{12} = A_{11} B_{12} + A_{12} B_{22} + \ldots + A_{1n} B_{n2}$

$C_{1q} = A_{11} B_{1q} + A_{12} B_{2q} + \ldots + A_{1n} B_{nq}$

$C_{21} = A_{21} B_{11} + A_{22} B_{21} + \ldots + A_{2n} B_{n1}$

$C_{22} = A_{21} B_{12} + A_{22} B_{22} + \ldots + A_{2n} B_{n2}$

$C_{2q} = A_{21} B_{1q} + A_{22} B_{2q} + \ldots + A_{2n} B_{nq}$

$C_{m1} = A_{m1} B_{11} + A_{m2} B_{21} + \ldots + A_{mn} B_{n1}$

$C_{m2} = A_{m1} B_{12} + A_{m2} B_{22} + \ldots + A_{mn} B_{n2}$

$C_{mq} = A_{m1} B_{1q} + A_{m2} B_{2q} + \ldots + A_{mn} B_{nq}$

- Example:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \qquad B = \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix}$$

$$C = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

9.3 ALGORITHM

Input: The order and elements of Matrix A and Matrix B.

Output: To compute product of Matrix A and Matrix B.

Step 1: Start

Step 2: Read the **order** of **matrix A: m_row and n_col**

Step 3: Read the elements into **matrix A (row major order): a [m_row] [n_col]**

Step 4: Display Matrix A: a [m_row] [n_col]

Step 5: Read the **order** of **matrix B: p_row and q_col**

Step 6: Read the elements into **matrix B (col major order): b [p_row] [q_col]**

Step 7: Display Matrix B: b [p_row] [q_col]

Step 8: Check **if n_col is not equal to p_row**. If true **goto Step 9** otherwise **goto Step 10**

Step 9: Display matrix multiplication not possible and goto Step 18

Step 10: Initialize:

row = 0

col = 0

k = 0

Step 11: Check **if row is less than m_row.** If true **goto Step 12** otherwise **goto Step 17**

Step 12: Check **if col is less than q_col**. If true initialize **c [row] [col] = 0** and **goto Step 13** otherwise **goto Step 16**

Step 13: Check **if k is less than n_col**. If true **goto Step 14** otherwise **goto Step 15**

Step 14: Compute:

c [row] [col] = c [row] [col] + (a [row] [k] * b [k] [col])

Increment k **goto Step 13**

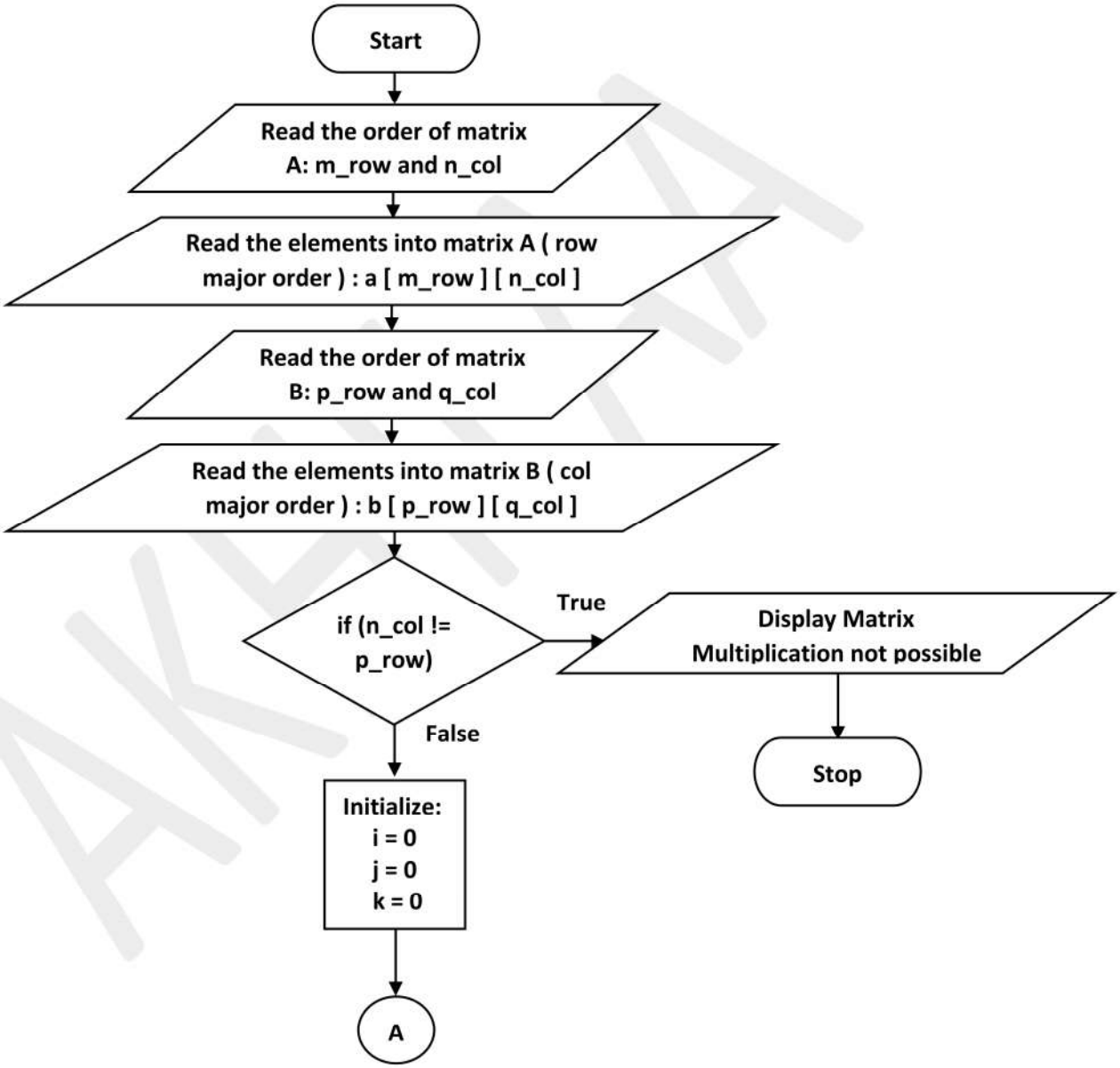
Step 15: Increment col **goto Step 12**

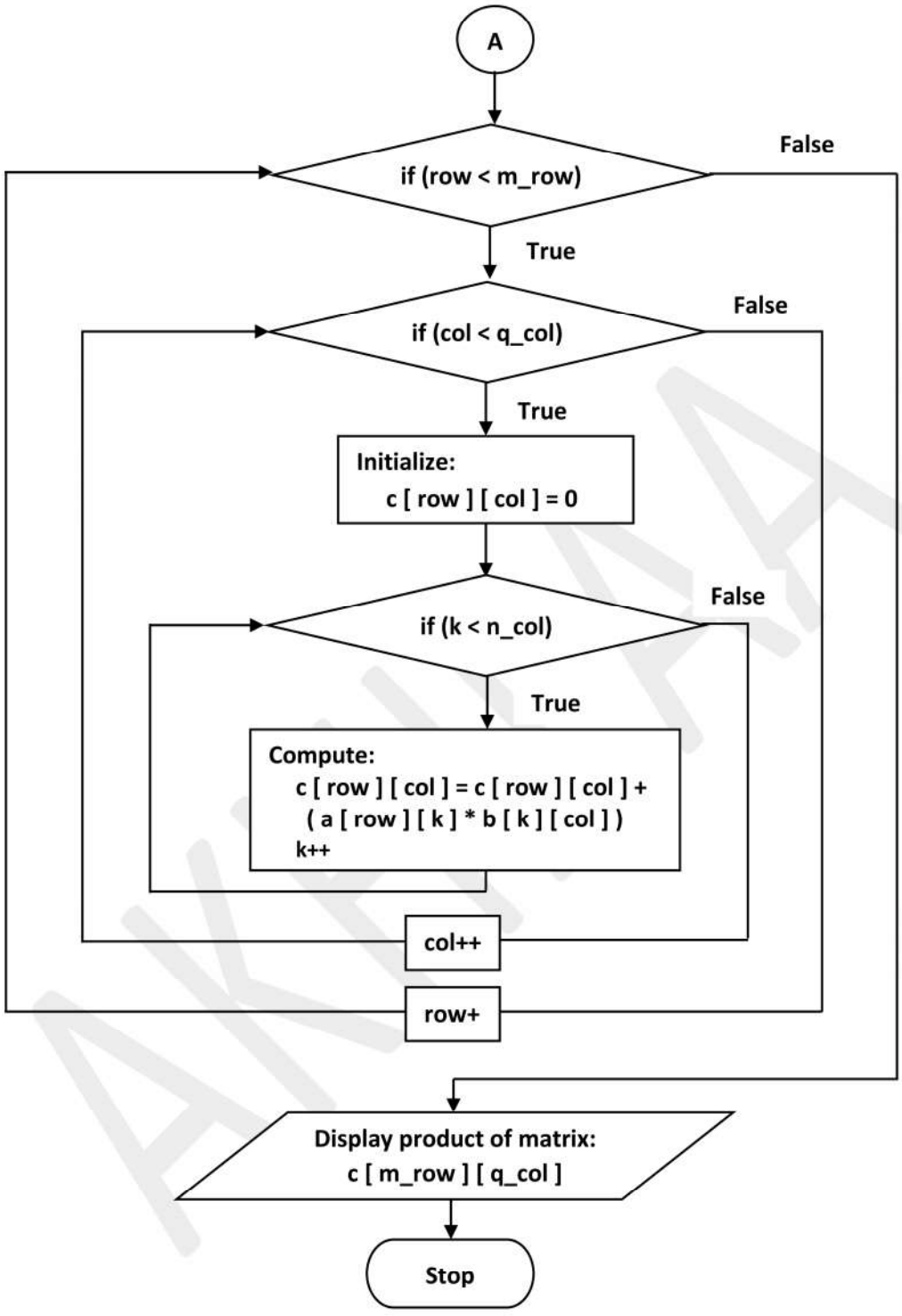
Step 16: Increment row **goto Step 11**

Step 17: Display the product of matrix: c [m_row] [q_col]

Step 18: Stop

9.4 FLOWCHART




A
if (row < m_row)
False
True
if (col < q_col)
False
True
Initialize:
c [row] [col] = 0
if (k < n_col)
False
True
Compute:
c [row] [col] = c [row] [col] +
(a [row] [k] * b [k] [col])
k++
col++
row+
Display product of matrix:
c [m_row] [q_col]
Stop

9.5 PROGRAM CODE

```
/* Program to perform matrix multiplication. */

#include<stdio.h>

int main()
{
        int m_row, n_col;
        int p_row, q_col;
        int row, col, k;
        int a [ 10 ] [ 10 ], b [ 10 ] [ 10 ], c [ 10 ] [ 10 ];

        // Input the order of matrix A ( m X n ).
        printf ( "\nEnter the order of matrix A ( row x col ): " );
        scanf ( "%d%d", &m_row, &n_col );

        // Input the elements into matrix A ( row major ).
        printf ( "\nEnter elements in row major order:\n" );
        for ( row = 0 ; row < m_row ; row++ )                    // iterate over the rows.
        {
                for ( col = 0 ; col < n_col ; col++ )            // for the current row,
                                                                 // iterate over the cols.
                {
                        scanf ( "%d", &a [ row ] [ col ] );
                }
        }

        // Display matrix A.
        printf ( "\nThe matrix A is:\n" );
        for ( row = 0 ; row < m_row ; row++ )
        {
                for ( col = 0 ; col < n_col ; col++ )
                {
                        printf ( "\t%d", a [ row ] [ col ] );
                }
                printf ( "\n" );
        }

        // Input the order of matrix B ( p X q ).
        printf ( "\nEnter the order of matrix B ( row x col ): " );
        scanf ( "%d%d", &p_row , &q_col );

        // Input the elements into matrix B ( col major ).
        printf ( "\nEnter elements in col major order:\n" );
        for ( col = 0 ; col < q_col ; col++ )                    // iterate over the cols.
        {
```

```
        for ( row = 0 ; row < p_row ; row++ )          // for the current col,
                                                       // iterate over the rows.
        {
                scanf ( "%d", &b [row] [col] );
        }
}

// Display matrix B.
printf ( "\nThe matrix B is:\n" );
for ( row = 0 ; row < p_row ; row++ )
{
        for ( col = 0 ; col < q_col ; col++ )
        {
                printf ( "\t%d", b [ row ] [ col ] );
        }
        printf ( "\n" );
}

// For multiplication of two matrices, the number of cols in first matrix
// should be equal to number of rows in the second matrix.
if ( n_col != p_row )
{
        printf("\nMatrix multiplication not possible!\n");
        return 0;
}

for ( row = 0 ; row < m_row ; row++ )              // iterate over the rows.
{
        for ( col = 0 ; col < q_col ; col++ )      // for the current row,
                                                   // iterate over the cols.
        {
                c [ row ] [ col ] = 0;
                for ( k = 0 ; k < n_col ; k++ )
                {
                        // multiply row of matrix A with col of matrix B.
                        c [ row ] [ col ] = c [ row ] [ col ] + ( a [ row ] [ k ] * b [ k ]
                        [ col ] );
                }
        }
}

// Display product of two matrices.
printf ( "\nThe Product ( A X B ) is:\n" );
for ( row = 0 ; row < m_row ; row++ )
{
        for ( col = 0 ; col < q_col ; col++ )
        {
```

```
                        printf ( "\t%d", c [ row ] [ col ] );
                }
                printf ( "\n" );
        }

        return 0;
}
```

9.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit matrix_multiplication.c
akhilaa@akhilaa-VirtualBox:~$ cc matrix_multiplication.c
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the order of matrix A ( row x col ): 2 2

Enter elements in row major order:
1 2 3 4

The matrix A is:
  1       2
  3       4

Enter the order of matrix B ( row x col ): 2 2

Enter elements in col major order:
1 2 3 4

The matrix B is:
  1       3
  2       4

The Product ( A X B ) is:
  5       11
  11      25

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the order of matrix A ( row x col ): 3 2

Enter elements in row major order:
6 5 4 3 2 1

The matrix A is:
  6       5
  4       3
  2       1
```

```
Enter the order of matrix B ( row x col ): 2 3

Enter elements in col major order:
1 2 3 4 5 6

The matrix B is:
  1       3       5
  2       4       6

The Product ( A X B ) is:
  16      38      60
  10      24      38
  4       10      16

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the order of matrix A ( row x col ): 2 3

Enter elements in row major order:
5 8 6 8 9 6

The matrix A is:
  5       8       6
  8       9       6

Enter the order of matrix B ( row x col ): 2 3

Enter elements in col major order:
1 8 6 2 3 5

The matrix B is:
  1       6       3
  8       2       5

Matrix multiplication not possible!

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the order of matrix A ( row x col ): 3 3

Enter elements in row major order:
1 99 2 88 3 77 4 66 5

The matrix A is:
  1       99      2
  88      3       77
  4       66      5
```

```
Enter the order of matrix B ( row x col ): 3 3

Enter elements in col major order:
6 55 7 44 8 33 9 22 10

The matrix B is:
  6       44      9
  55      8       22
  7       33      10

The Product ( A X B ) is:
  5465    902     2207
  1232    6437    1628
  3689    869     1538

akhilaa@akhilaa-VirtualBox:~$
```

9.7 VIVA QUESTIONS

- What is a 2Dimensional array?
- What are the different types of arrays?
- How do you declare a 2D array?
- What is multi-dimensional array?
- What is row major order?
- What is column major order?

10 BINARY SEARCH (VTU-PROGRAM 8)

10.1 PROBLEM STATEMENT

Develop, implement and execute a C program to search a Name in a list of names using Binary searching Technique.

10.2 PLAN OF SOLUTION

A binary search algorithm finds the position of a target value within a sorted array.

A binary search divides a range of values into halves, and continues to narrow down the field of search until the unknown value is found. It is the classic example of a "divide and conquer" algorithm.

The binary search algorithm begins by comparing the target value to the value of the middle element of the sorted array. If the target value is equal to the middle element's value, then the position is returned and the search is finished. If the target value is less than the middle element's value, then the search continues on the lower half of the array; or if the target value is greater than the middle element's value, then the search continues on the upper half of the array. This process continues, eliminating half of the elements, and comparing the target value to the value of the middle element of the remaining elements - until the target value is either found (and its associated element position is returned), or until the entire array has been searched (and "not found" is returned).

For example:

Sorted array: L = [1, 3, 4, 6, 8, 9, 11]

Target value: X = 4

Compare X to 6. X is smaller. Repeat with L = [1, 3, 4].

Compare X to 3. X is larger. Repeat with L = [4].

Compare X to 4. X equals 4, so the position is returned.

10.3 ALGORITHM

Input: A sorted array of strings.

Output: To search for a string using Binary Search.

Step 1: **Start**

Step 2: Read the number of strings: **n**

Step 3: Read the strings: **s [n]**

Step 4: **Sort the strings using Bubble Sort**

Step 5: **Display the strings in sorted order: s [n]**

Step 6: Read the string to be searched: **search_string**

Step 7: Initialize:

first = 0

last = n – 1

Step 8: Check **while first is less than or equal to last**. If true **goto Step 9** otherwise **goto Step 15**

Step 9: Compute:

mid = (first + last) / 2

res = strcmp (s [mid] , search_string)

Step 10: Check **if res is equal to zero**. If true **goto step 11** otherwise **goto Step 12**

Step 11: Display String found at position: mid and goto Step 16

Step 12: Check **if res is greater than zero**. If true **goto Step 13** otherwise **goto Step 14**

Step 13: Compute:

last = mid – 1 **goto Step 8**

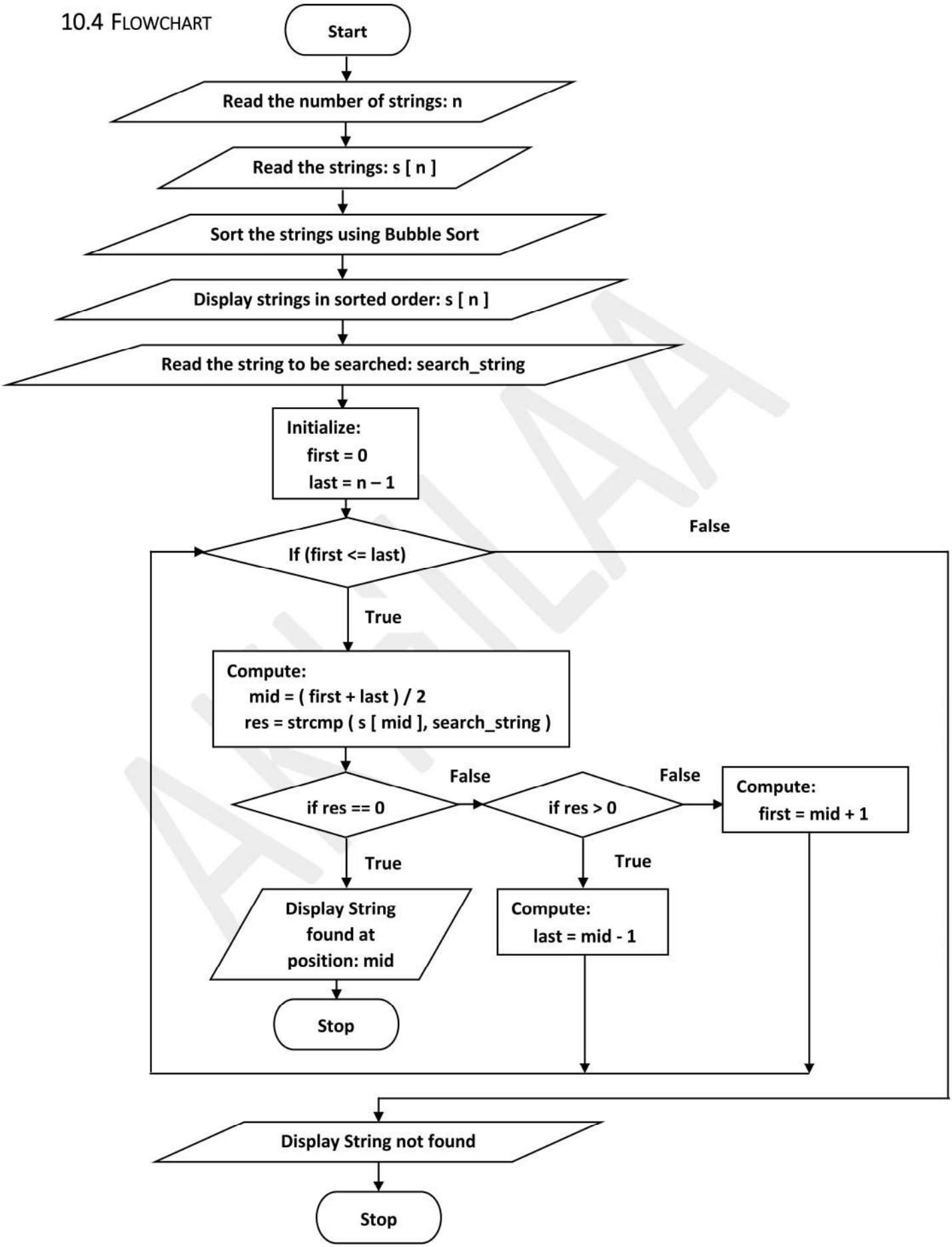
Step 14: Compute:

first = mid + 1 **goto Step 8**

Step 15: Display String Not Found

Step 16: Stop

10.4 FLOWCHART



10.5 PROGRAM CODE

```
/* Program to perform Binary Search on strings. */

#include<stdio.h>
#include<string.h>

int main()
{
        char s [ 20 ] [ 20 ], temp [ 20 ], search_string [ 20 ];
        int n, i, j, first, last, mid, res;

        //Input the number of strings.
        printf ( "\nEnter the number of strings: " );
        scanf ( "%d", &n );

        //Input the strings.
        printf ( "\nEnter the strings:\n" ) ;
        for ( i = 0 ; i < n ; i++ )
        {
                scanf ( "%s", s [ i ] );
        }

        // Sort strings using Bubble Sort.
        for ( i = 1 ; i < n ; i++ )
        {
                for ( j = 0 ; j < n - i ; j++ )
                {
                        if ( strcmp ( s [ j ], s [ j + 1 ] ) > 0 )
                        {
                                strcpy ( temp, s [ j ] );
                                strcpy ( s [ j ], s [ j + 1 ] );
                                strcpy ( s [ j + 1 ], temp );
                        }
                }
        }

        // Display the sorted strings.
        printf ( "\nStrings in sorted order are:\n" );
        for ( i = 0 ; i < n ; i++ )
        {
                printf( "%s\n", s [ i ] );
        }

        // Input the string that you want to search.
        printf ( "\nEnter string to search: " );
        scanf ( "%s", search_string );
```

```
        // Search for string using Binary Search.
        first = 0;
        last = n-1;

        while ( first <= last )
        {
                mid = ( first + last ) / 2;

                res = strcmp ( s [ mid ], search_string );

                // The string in mid position and search string are same.
                if ( res == 0 )
                {
                        printf ( "\n\nString found at %d position\n\n", mid );
                        return 0;
                }
                // The string in mid position is greater than search string.
                // Search in the first half of the array.
                else if ( res > 0 )
                {
                        last = mid - 1;
                }
                // The string in mid position is less than search string.
                // Search in the second half of the array.
                else
                {
                        first = mid + 1;
                }

        }

        printf ( "\nString not found\n\n" );

        return 0;
}
```

10.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit binary_search.c
akhilaa@akhilaa-VirtualBox:~$ cc binary_search.c
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number of strings: 5

Enter the strings:
hello
```

```
good
day
bye
cmrit

Strings in sorted order are:
bye
cmrit
day
good
hello

Enter string to search: cmrit

String found at 1 position

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number of strings: 5

Enter the strings:
hello
bye
day
cmrit
good

Strings in sorted order are:
bye
cmrit
day
good
hello

Enter string to search: cmrims

String not found

akhilaa@akhilaa-VirtualBox:~$ ./a.out
```

10.7 ALTERNATE PROGRAM CODE

```
/* Program to perform Binary Search on strings. */

#include<stdio.h>
#include<string.h>

int main()
```

```
{
        char s [ 20 ] [ 20 ], temp [ 20 ], search_string [ 20 ];
        int n, i, j, first, last, mid, res;

        //Input the number of strings.
        printf ( "\nEnter the number of strings: " );
        scanf ( "%d", &n );

        //Input the strings.
        printf ( "\nEnter the strings in sorted order:\n" ) ;
        for ( i = 0 ; i < n ; i++ )
        {
                scanf ( "%s", s [ i ] );
        }

        // Input the string that you want to search.
        printf ( "\nEnter string to search: " );
        scanf ( "%s", search_string );

        // Search for string using Binary Search.
        first = 0;
        last = n-1;

        while ( first <= last )
        {
                mid = ( first + last ) / 2;

                res = strcmp ( s [ mid ], search_string );

                // The string in mid position and search string are same.
                if ( res == 0 )
                {
                        printf ( "\nString found at %d position\n\n", mid );
                        return 0;
                }
                // The string in mid position is greater than search string.
                // Search in the first half of the array.
                else if ( res > 0 )
                {
                        last = mid - 1;
                }
                // The string in mid position is less than search string.
                // Search in the second half of the array.
                else
                {
                        first = mid + 1;
                }
```

```
        }

        printf ( "\nString not found\n\n" );

        return 0;
}
```

Output:

```
akhilaa@akhilaa-VirtualBox:~$ gedit binary_search.c
akhilaa@akhilaa-VirtualBox:~$ cc binary_search.c
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number of strings: 5

Enter the strings in sorted order:
hello
good
day
bye
cmrit

Enter string to search: cmrit

String found at 1 position

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number of strings: 5

Enter the strings in sorted order:
hello
bye
day
cmrit
good

Enter string to search: cmrims

String not found

akhilaa@akhilaa-VirtualBox:~$ ./a.out
```

10.8 VIVA QUESTIONS

- How does strcmp() work?
- What is strcpy()?

11 STRING COPY (VTU-PROGRAM 9A)

11.1 PROBLEM STATEMENT

Write and execute a C program that implements string copy operation STRCOPY (str1, str2) that copies a string str1 to another string str2 without using library function.

11.2 PLAN OF SOLUTION

- Strings are array of characters terminated by a null character '\0'.
- The aim here is given string1, copy the contents of string1 to string2 without using built-in function.
- Initialize index to 0.
- Copy individual characters from string1 to string2 until null character is encountered.
- Increment the index value.
- Once all the characters are copied to string2 append a null character at the end of string2.

11.3 ALGORITHM

Input: Two strings - str1 and str2.

Output: To copy str1 into str2.

Step 1: Start

Step 2: Read a String to copy: **str1**

Step 3: Compute:

strcopy (str1, str2)

Step 4: Display Copying Success: str2

Step 5: Stop

VOID STRCOPY (CHAR STR1 [ ], CHAR STR2 [ ])

Step 1: Initialize:

i = 0

Step 2: Check **while str1 [i] is not equal to '\0'**. If true **goto Step 3**

otherwise **goto Step 4**

Step 3: Compute:

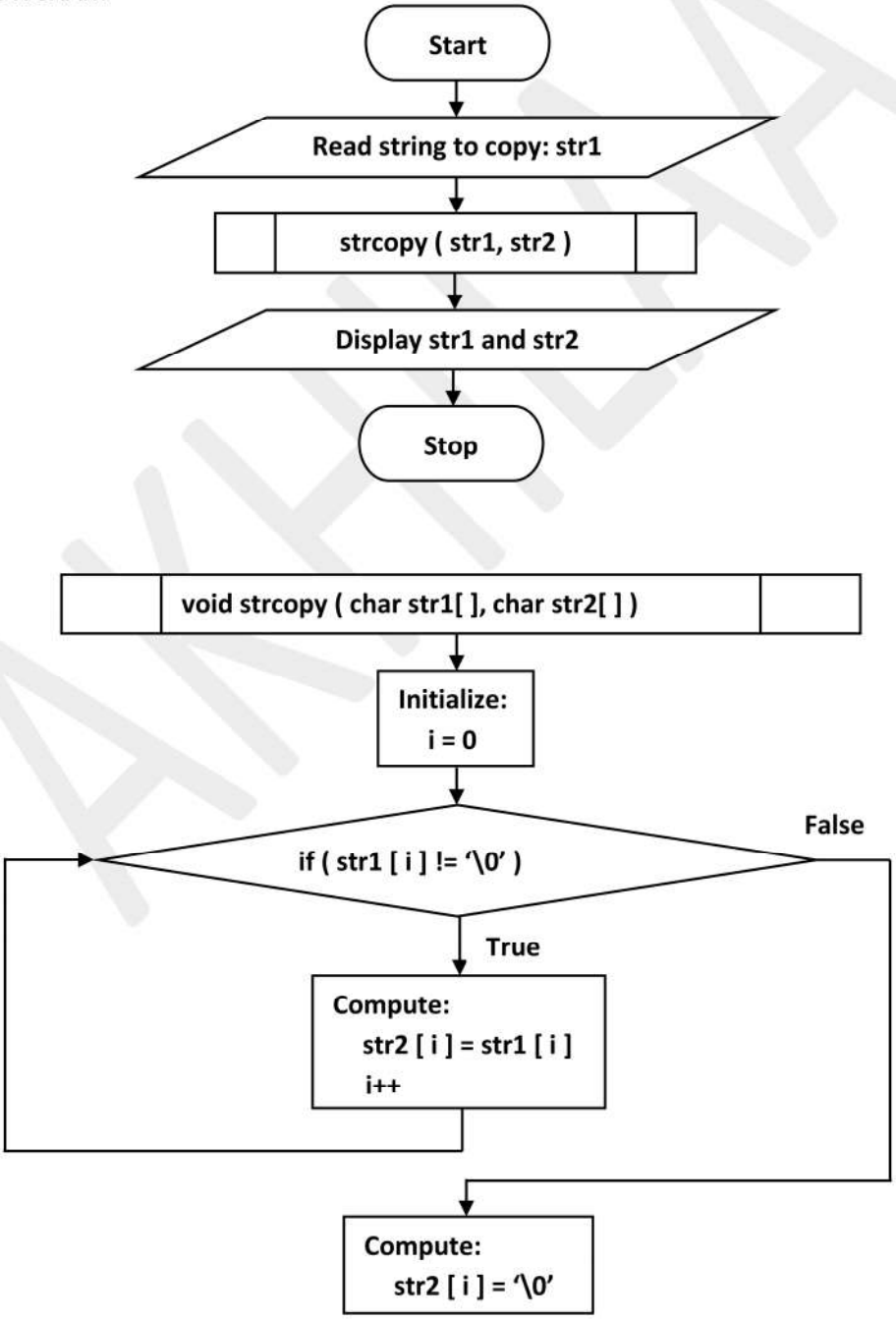
str2 [i] = str1 [i]

Increment I **goto Step 3**

Step 4: Compute:

str2 [i] = '\0'

11.4 FLOWCHART



11.5 PROGRAM CODE

/* Program to create a function to copy a string to another string without using built-in function. */

```
#include<stdio.h>

// Function declaration for strcopy.
void strcopy ( char str1 [ ], char str2 [ ] );

int main()
{
        char str1 [ 20 ], str2 [ 20 ];

        // Input the string that you want to copy.
        printf ( "\nEnter string to copy: " );
        gets ( str1 );

        // Function call.
        strcopy ( str1 , str2 );

        // Display the contents of str1 and str2.
        printf ( "\n\nCopying success!!!!!\n" );
        printf ( "\nThe first string is: " );
        puts ( str1 );
        printf ( "\nThe second string is: " );
        puts ( str2 );

        return 0;
}

// Function definition for strcopy.
void strcopy ( char str1 [ ], char str2 [ ] )
{
        int i;

        // Copying the contents of str1 to str2 until NULL is encountered.
        i = 0;
        while ( str1 [ i ] != '\0' )
        {
                str2 [ i ] = str1 [ i ];
                i++;
        }

        // Append NULL character at the end of str2.
        str2 [ i ] = '\0';
}
```

11.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit strcopy.c
akhilaa@akhilaa-VirtualBox:~$ cc strcopy.c
strcopy.c: In function ‘main’:
strcopy.c:31:2: warning: ‘gets’ is deprecated (declared at /usr/include/stdio.h:638) [-
Wdeprecated-declarations]
 gets ( str1 );
 ^
/tmp/ccCodl05.o: In function `main':
strcopy.c:(.text+0x6d): warning: the `gets' function is dangerous and should not be
used.
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter string to copy: welcome to cmrit

Copying success!!!!!

The first string is: welcome to cmrit

The second string is: welcome to cmrit

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter string to copy: information science

Copying success!!!!!

The first string is: information science

The second string is: information science

akhilaa@akhilaa-VirtualBox:~$
```

11.7 ALTERNATE PROGRAM CODE

/* Program to create a function to copy a string to another string without using built-in function. */

```
#include<stdio.h>

// Function declaration for strcopy.
void strcopy ( char str1 [ ], char str2 [ ] );

int main()
{
        char str1 [ 20 ], str2 [ 20 ];
```

```
        // Input the string that you want to copy.
        printf ( "\nEnter string to copy: " );
        gets ( str1 );

        // Function call.
        strcopy ( str1 , str2 );

        // Display the contents of str1 and str2.
        printf ( "\n\nCopying success!!!!!\n" );
        printf ( "\nThe first string is: " );
        puts ( str1 );
        printf ( "\nThe second string is: " );
        puts ( str2 );

        return 0;
}

// Function definition for strcopy.
void strcopy ( char str1 [ ], char str2 [ ] )
{
        int i;

        // Copying the contents of str1 to str2 until NULL is encountered.
        for ( i = 0; str1 [ i ] != '\0'; i++ )
        {
                str2 [ i ] = str1 [ i ];
        }

        // Append NULL character at the end of str2.
        str2 [ i ] = '\0';
}
```

11.8 VIVA QUESTIONS

- How are strings?
- How do u declare a string?
- What is a function?
- What are actual parameters and formal parameters?
- What are the different types of functions?
- What do u mean by a void fun()?
- What are the different string manipulation functions?

12 FREQUENCY OF VOWELS AND CONSONANTS (VTU-PROGRAM 9B)

12.1 PROBLEM STATEMENT

Write and execute a C program that reads a sentence and print frequency of vowels and total count of consonants.

12.2 PLAN OF SOLUTION

- The aim here is given a string, count the number of vowels, consonants and white spaces.
- Check if each individual character in string is either a vowel, consonant or whitespace and increment.

12.3 ALGORITHM

Input: A sentence.

Output: To find the frequency of vowels, consonants and spaces.

Step 1: Start

Step 2: Read a String: **string**

Step 3: Initialize:

vow = 0

con = 0

tab = 0

i = 0

Step 4: Check **while str [i] is not equal to NULL**. If true **goto Step 5** otherwise **goto Step 10**

Step 5: Check **if str [i] is either 'a', 'e', 'i', 'o' or 'u'**. If true **goto Step 6** otherwise **goto Step 7**

Step 6: Compute:

Increment vow

Increment i **goto Step 4**

Step 7: Check **if str [i] is a single space or a tab space**. If true **goto Step 8** otherwise **goto Step 9**

Step 8: Compute:

Increment tab

Increment i **goto Step 4**

Step 9: Compute:

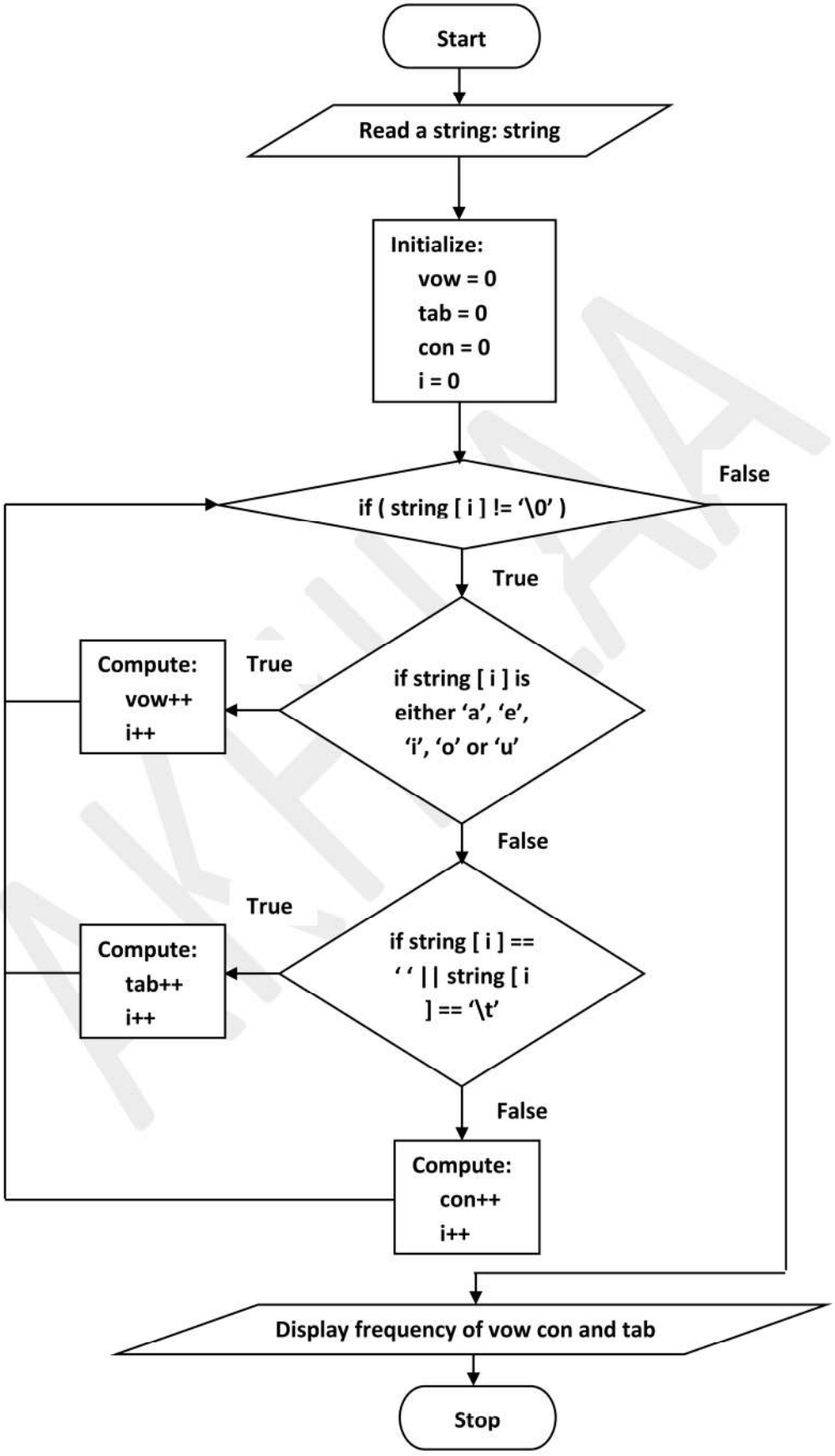
Increment con

Increment i **goto Step 4**

Step 10: Display the Frequency of Vowels, Consonants and Spaces: vow, con, tab

Step 11: Stop

12.4 FLOWCHART



12.5 PROGRAM CODE

```
/* Program to find the frequency of vowels and consonants. */

#include<stdio.h>

int main()
{
        char string [ 20 ];
        int vow = 0, con = 0, tab = 0, i = 0;

        // Input the string.
        printf ( "\nEnter the string: " );
        gets ( string );

        while ( string [ i ] != '\0' )
        {
                // Check if the string [ i ] contains vowels a, e, i, o, u. Increment
                // vowel count.
                if ( string [ i ] == 'a' || string [ i ] == 'e' || string [ i ] == 'i' || string [ i ]
                =='o' || string [ i ] == 'u' )
                {
                        vow++;
                        i++;
                }
                // Check if the string [ i ] contains a single space or a tab space.
                // Increment tab count.
                else if ( string [ i ] == ' ' || string [ i ] == '\t' )
                {
                        tab++;
                        i++;
                }
                // Check if the string [ i ] contains consonants. Increment con count.
                else
                {
                        con++;
                        i++;
                }
        }

        // Display the frequency of vowels, consonants and spaces.
        printf("\n Vowels = %d \n Consonants = %d \n Spaces = %d
        \n\n",vow,con,tab);

        return 0;
}
```

12.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit frequency_vowcons.c
akhilaa@akhilaa-VirtualBox:~$ cc frequency_vowcons.c
frequency_vowcons.c: In function ‘main’:
frequency_vowcons.c:10:2: warning: ‘gets’ is deprecated (declared at
/usr/include/stdio.h:638) [-Wdeprecated-declarations]
  gets ( string );
  ^
/tmp/cccUt6DE.o: In function `main':
frequency_vowcons.c:(.text+0x49): warning: the `gets' function is dangerous and
should not be used.
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the string: hi welcome to cmrit

 Vowels = 6
 Consonants = 10
 Spaces = 3

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the string: information science

 Vowels = 8
 Consonants = 10
 Spaces = 1
```

12.7 ALTERNATE PROGRAM CODES

- **/* Program to find the frequency of vowels and consonants. */**

```
#include<stdio.h>

int main()
{
        char string [ 20 ];
        int vow = 0, con = 0, tab = 0, i = 0;

        // Input the string.
        printf ( "\nEnter the string: " );
        gets ( string );

        for ( i = 0; string [ i ] != '\0'; i++ )
        {
                // Check if the string [ i ] contains vowels a, e, i, o, u. Increment
                // vowel count.
```

```
                if ( string [ i ] == 'a' || string [ i ] == 'e' || string [ i ] == 'i' || string [ i ]
                =='o' || string [ i ] == 'u' )
                {
                        vow++;
                }
                // Check if the string [ i ] contains a single space or a tab space.
                 // Increment tab count.
                else if ( string [ i ] == ' ' || string [ i ] == '\t' )
                {
                        tab++;
                }
                // Check if the string [ i ] contains consonants. Increment con count.
                else
                {
                        con++;
                }
        }

        // Display the frequency of vowels, consonants and spaces.
        printf("\n Vowels = %d \n Consonants = %d \n Spaces = %d
        \n\n",vow,con,tab);

        return 0;
}
```

- **/* Program to find the frequency of vowels and consonants. */**

```
#include<stdio.h>

int main()
{
        char string [ 20 ];
        int i, ac = 0, ec = 0, ic = 0, oc = 0, uc = 0, tab = 0, con = 0;

        // Input the string.
        printf ( "\nEnter the string: " );
        gets ( string );

        for ( i = 0; string [ i ] != '\0'; i++ )
        {
                // Check if the string [ i ] contains vowels a. Increment a count.
                if ( string [ i ] == 'a' || string [ i ] == 'A' )
                {
                        ac++;
                }
                // Check if the string [ i ] contains vowels e. Increment e count.
                else if ( string [ i ] == 'e' || string [ i ] == 'E' )
```

```
                {
                        ec++;
                }
                // Check if the string [ i ] contains vowels i. Increment i count.
                else if ( string [ i ] == 'i' || string [ i ] == 'I' )
                {
                        ic++;
                }
                // Check if the string [ i ] contains vowels o. Increment o count.
                else if ( string [ i ] == 'o' || string [ i ] == 'O' )
                {
                        oc++;
                }
                // Check if the string [ i ] contains vowels u. Increment u count.
                else if ( string [ i ] == 'u' || string [ i ] == 'U' )
                {
                        uc++;
                }
                // Check if the string [ i ] contains a single space or a tab space.
                // Increment tab count.
                else if ( string [ i ] == ' ' || string [ i ] == '\t' )
                {
                        tab++;
                }
                // Check if the string [ i ] contains consonants. Increment con count.
                else
                {
                        con++;
                }
        }

        // Display the frequency of vowels, consonants and spaces.
        printf ( "\n a = %d \n e = %d \n i = %d \n o = %d \n u =%d \n Consonants = %d
        \n Spaces = %d \n\n", ac, ec, ic, oc, uc, con, tab);

        return 0;
}
```

- **/* Program to find the frequency of vowels and consonants. */**

```
#include<stdio.h>

int main()
{
        char string [ 20 ];
        int i, ac = 0, ec = 0, ic = 0, oc = 0, uc = 0, tab = 0, con = 0;
```

```
        // Input the string.
        printf ( "\nEnter the string: " );
        gets ( string );

        for ( i = 0; string [ i ] != '\0'; i++ )
        {
                switch ( string [ i ] )
                {
                        case 'a':
                        case 'A': ac++;
                                  break;

                        case 'e':
                        case 'E': ec++;
                                  break;

                        case 'i':
                        case 'I': ic++;
                                  break;

                        case 'o':
                        case 'O': oc++;
                                  break;

                        case 'u':
                        case 'U': uc++;
                                  break;

                        case ' ':
                        case '\t': tab++;
                                  break;

                        default: con++;
                }
        }

        // Display the frequency of vowels, consonants and spaces.
        printf ( "\n a = %d \n e = %d \n i = %d \n o = %d \n u =%d \n Consonants = %d
        \n Spaces = %d \n\n", ac, ec, ic, oc, uc, con, tab);

        return 0;
}
```

Output:

```
akhilaa@akhilaa-VirtualBox:~$ gedit frequency_vowcons.c
akhilaa@akhilaa-VirtualBox:~$ cc frequency_vowcons.c
```

```
frequency_vowcons.c: In function ‘main’:
frequency_vowcons.c:12:2: warning: ‘gets’ is deprecated (declared at
/usr/include/stdio.h:638) [-Wdeprecated-declarations]
  gets ( string );
^
/tmp/ccK6isjv.o: In function `main':
frequency_vowcons.c:(.text+0x61): warning: the `gets' function is dangerous and
should not be used.
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the string: HI WELCOME TO CMRIT

 a = 0
e = 2
 i = 2
 o = 2
u =0
Consonants = 10
 Spaces = 3

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the string: INFORMATION SCIENCE

a = 1
 e = 2
 i = 3
 o = 2
 u =0
 Consonants = 10
 Spaces = 1

akhilaa@akhilaa-VirtualBox:~$
```

- **/* Program to find the frequency of vowels and consonants. */**

```
#include<stdio.h>

int main()
{
        char string [ 20 ], ch;
        int i, ac = 0, ec = 0, ic = 0, oc = 0, uc = 0, tab = 0, con = 0;

        // Input the string.
        printf ( "\nEnter the string: " );
        gets ( string );
```

```
        for ( i = 0; string [ i ] != '\0'; i++ )
        {
                ch = tolower ( string [ i ] );
                switch ( ch )
                {
                        case 'a': ac++;
                                  break;

                        case 'e': ec++;
                                  break;

                        case 'i': ic++;
                                  break;

                        case 'o': oc++;
                                  break;

                        case 'u': uc++;
                                  break;

                        case ' ':
                        case '\t': tab++;
                                   break;

                        default: con++;
                }
        }

        // Display the frequency of vowels, consonants and spaces.
        printf ( "\n a = %d \n e = %d \n i = %d \n o = %d \n u =%d \n Consonants = %d
        \n Spaces = %d \n\n", ac, ec, ic, oc, uc, con, tab);

        return 0;
}
```

Output:

```
akhilaa@akhilaa-VirtualBox:~$ gedit frequency_vowcons.c
akhilaa@akhilaa-VirtualBox:~$ cc frequency_vowcons.c
frequency_vowcons.c: In function ‘main’:
frequency_vowcons.c:12:2: warning: ‘gets’ is deprecated (declared at
/usr/include/stdio.h:638) [-Wdeprecated-declarations]
  gets ( string );
  ^
/tmp/ccK6isjv.o: In function `main':
frequency_vowcons.c:(.text+0x61): warning: the `gets' function is dangerous and
```

should not be used.

```
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the string: HI WELCOME TO CMRIT

 a = 0
e = 2
i = 2
o = 2
u =0
Consonants = 10
 Spaces = 3

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the string: INFORMATION SCIENCE

a = 1
e = 2
 i = 3
o = 2
u =0
Consonants = 10
Spaces = 1

akhilaa@akhilaa-VirtualBox:~$
```

- **/* Program to find the frequency of vowels and consonants. */**

```
#include<stdio.h>

int main()
{
        char string [ 20 ];
        int vow = 0, con = 0, tab = 0, i = 0;

        // Input the string.
        printf ( "\nEnter the string: " );
        gets ( string );

        while ( string [ i ] != '\0' )
        {
                // Check if the string [ i ] contains vowels a, e, i, o, u. Increment
                // vowel count.
                ch = tolower ( string [ i ] );
                if ( ch == 'a' || ch == 'e' || ch == 'i' || ch =='o' || ch == 'u' )
                {
```

```
                vow++;
                i++;
            }
            // Check if the string [ i ] contains a single space or a tab space.
            // Increment tab count.
            else if ( ch == ' ' || ch == '\t' )
            {
                tab++;
                i++;
            }
            // Check if the string [ i ] contains consonants. Increment con count.
            else if ( ch >= 'a' && ch >= 'z' )
            {
                con++;
                i++;
            }
        }

        // Display the frequency of vowels, consonants and spaces.
        printf("\n Vowels = %d \n Consonants = %d \n Spaces = %d \n\n", vow, con,
        tab);

        return 0;
    }
```

12.8 VIVA QUESTIONS

- How does switch statement work?
- What do you mean by break?
- What will happen if you don't use break after each case in switc?

13 CIRCULAR RIGHT SHIFT (VTU-PROGRAM 10A)

13.1 PROBLEM STATEMENT

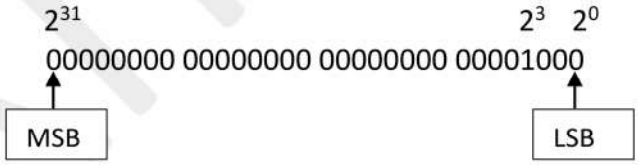
Design and develop a C function RightShift (x , n) that takes two integers x and n as input and returns value of the integer x rotated to the right by n positions. Assume the integers are unsigned. Write a C program that invokes this function with different values for x and n and tabulate the results with suitable headings.

13.2 PLAN OF SOLUTION

- The right shift operator (>>) shifts the bit pattern by the specified number of positions.
- For example: Suppose we want to perform circular-right-shift on 8 by 4 times.
  - First convert the given number 8 into binary format.

$$(8)_2 = \underset{2^3}{1}\ \underset{2^2}{0}\ \underset{2^1}{0}\ \underset{2^0}{0}$$

  - Since we are using a 32-bit machine the binary representation of the number would be



  - When right shift operator is applied to the above bit pattern, shifting of bit starts from Most Significant Bit (MSB).
  - When the above number is shifted one time (8 >> 1) then the result would be 4.

00000000 00000000 00000000 00001000 → Before Shift

00000000 00000000 00000000 00000100 → After Shift

2^{31} 2^2 2^0

The final result would be: (8 >> 1) = 2^2 = 4

- When the above number is shifted two times (8 >> 2) then the result would be 2.

00000000 00000000 00000000 00001000 → Before Shift

00000000 00000000 00000000 00000010 → After Shift

2^{31} $2^1 2^0$

The final result would be: (8 >> 2) = 2^1 = 2

- When the above number is shifted three times (8 >> 3) then the result would be 1.

00000000 00000000 00000000 00001000 → Before Shift

00000000 00000000 00000000 00000001 → After Shift

2^{31} 2^0

The final result would be: (8 >> 3) = 2^0 = 1

- When the above number is shifted four times (8 >> 4) then the result would be 2147483648.

00000000 00000000 00000000 00001000 → Before Shift

00000000 00000000 00000000 00000000 → After Shift

Since the aim here is to perform circular right shift on the given number we have to explicitly add 1 to MSB. This is done in the program by adding 2^{31} to the number after performing right shift. This is as shown below:

10000000 00000000 00000000 00000000

2^{31} 2^0

The final result would be: (8 >> 4) = 2^{31} = 2147483648

13.3 ALGORITHM

Input: Two positive integers x (the number to be rotated) and n (the number of times x has to be rotated).

Output: To display the result after rotation.

Step 1: Start

Step 2: Read the number you want to shift: **x**

Step 3: Read the number of times you want to shift the number: **n**

Step 4: Compute:

result = right_shift (x, n)

Step 5: Display the result after rotating x by n times is: result

Step 6: Stop

UNSIGNED INT RIGHTSHIFT (UNSIGNED INT X, UNSIGNED INT N)

Step 1: Initialize:

i = 0

Step 2: Check **if i is less than n**. If true **goto Step 3** otherwise **goto Step 6**

Step 3: Check **if x mod 2 is equal to zero**. If true **goto Step 4** otherwise **goto Step 5**

Step 4: Compute:

x = x >> 1

Increment i **goto Step 2**

Step 5: Compute:

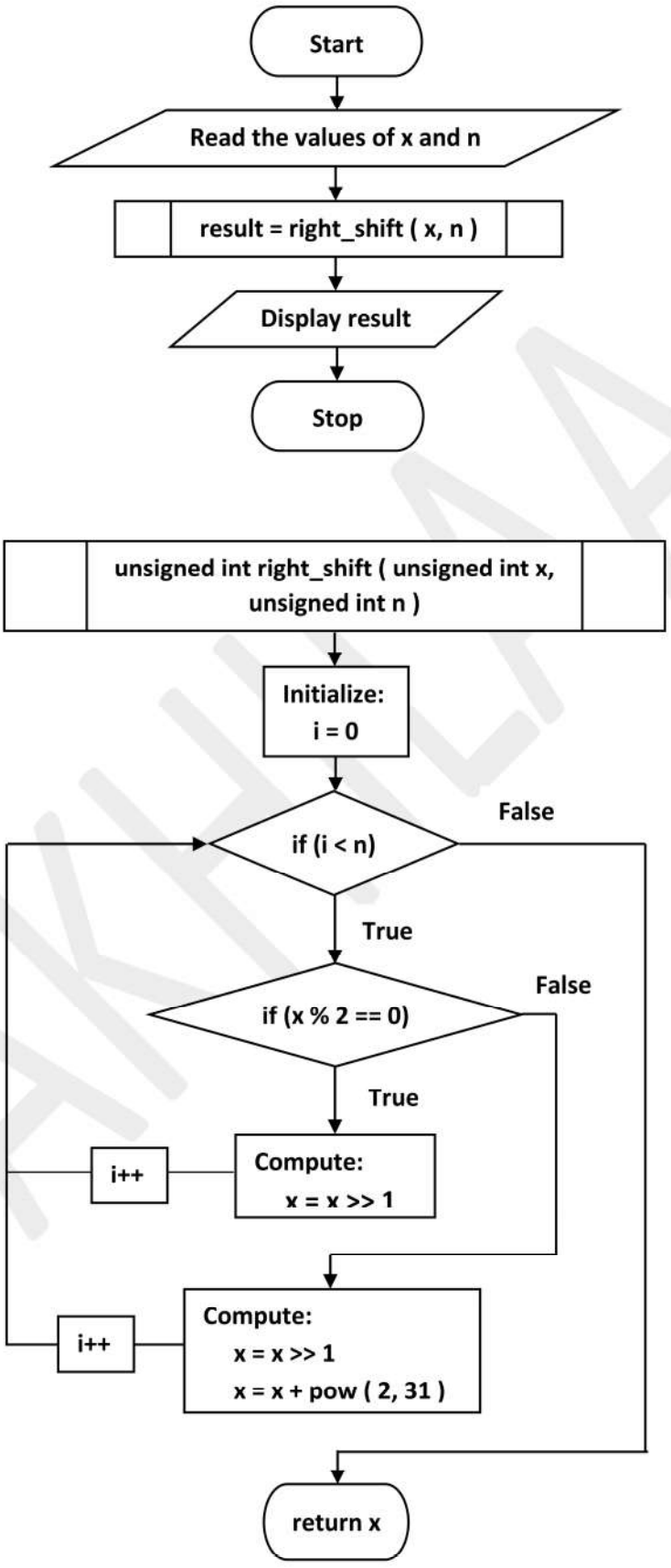
x = x >> 1

x = x + pow (2, 31)

Increment i **goto Step 2**

Step 6: Return the value of x

13.4 FLOWCHART



13.5 PROGRAM CODE

```
/* Program to perform circular right shift. */

#include<stdio.h>
#include<math.h>

// Function definition.
unsigned int right_shift ( unsigned int x, unsigned int n );

int main ()
{
        unsigned int x, n, result;

        // Input the number for which you want to perform and how many times
        // you want to shift.
        printf ( "\nEnter the number you want to shift: " );
        scanf ( "%u", &x );
        printf ( "\nEnter the number of times you want to shift the number: " );
        scanf ( "%u", &n );

        // Fuction call.
        result = right_shift ( x, n );

        // Display the result.
        printf ( "\nResult after shifting %u by %u bits is : %u\n\n", x, n, result );

        return 0;
}

// Function declaration.
unsigned int right_shift ( unsigned int x, unsigned int n )
{
        int i;

        for ( i = 0 ; i < n ; i++ )
        {
                if ( x % 2 == 0 )
                        x = x >> 1;       // only shift if most significant bit is zero.
                else
                {
                        x = x >> 1;       // circular shift if most significant bit is non-
                                          // zero.
                        x = x + pow ( 2, 31 );   // circular shift is done by adding
                                                 // pow(2,31) to the right shifted value.
                }
        }
```

```
        return x;
}
```

13.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit right_shift.c
akhilaa@akhilaa-VirtualBox:~$ cc right_shift.c -lm
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number you want to shift: 8

Enter the number of times you want to shift the number: 1

Result after shifting 8 by 1 bits is : 4

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number you want to shift: 8

Enter the number of times you want to shift the number: 2

Result after shifting 8 by 2 bits is : 2

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number you want to shift: 8

Enter the number of times you want to shift the number: 3

Result after shifting 8 by 3 bits is : 1

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number you want to shift: 8

Enter the number of times you want to shift the number: 4

Result after shifting 8 by 4 bits is : 2147483648

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number you want to shift: 8

Enter the number of times you want to shift the number: 5

Result after shifting 8 by 5 bits is : 1073741824

akhilaa@akhilaa-VirtualBox:~$ ./a.out
```

Enter the number you want to shift: 7

Enter the number of times you want to shift the number: 1

Result after shifting 7 by 1 bits is : 2147483651

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number you want to shift: 7

Enter the number of times you want to shift the number: 2

Result after shifting 7 by 2 bits is : 3221225473

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number you want to shift: 7

Enter the number of times you want to shift the number: 3

Result after shifting 7 by 3 bits is : 3758096384

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number you want to shift: 7

Enter the number of times you want to shift the number: 4

Result after shifting 7 by 4 bits is : 1879048192

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number you want to shift: 7

Enter the number of times you want to shift the number: 5

Result after shifting 7 by 5 bits is : 939524096

akhilaa@akhilaa-VirtualBox:~$

13.7 Alternate Program Code

/* Program to perform circular right shift. */

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

```
// Function definition.
unsigned int right_shift ( unsigned int x, unsigned int n );

int main ()
{
        unsigned int x, n, result;

        // Input the number for which you want to perform and how many times
        // you want to shift.
        printf ( "\nEnter the number you want to shift: " );
        scanf ( "%u", &x );
        printf ( "\nEnter the number of times you want to shift the number: " );
        scanf ( "%u", &n );

        // Fuction call.
        result = right_shift ( x, n );

        // Display the result.
        printf ( "\nResult after shifting %u by %u bits is : %u\n\n", x, n, result );

        return 0;
}

// Function declaration.
unsigned int right_shift ( unsigned int x, unsigned int n )
{
        int i;

        for ( i = 0 ; i < n ; i++ )
        {
                if ( x % 2 == 0 )
                        x = x >> 1;       // only shift if most significant bit is zero.
                else
                {
                        x = x >> 1;       // circular shift if most significant bit is non-
                                          // zero.
                        x = x + pow ( sizeof ( unsigned int ) * 8 -1 , 31 );       // circular
                        shift is done by adding pow(2,31) to the right shifted value.
                }
        }

        return x;
}
```

13.8 VIVA QUESTIONS

- What do you mean by unsigned int?
- How does right shift operator(>>) work?
- What is pow()?

14 PRIME NUMBER (VTU-PROGRAM 10B)

14.1 PROBLEM STATEMENT

Design and develop a C function isprime (num) that accepts an integer argument and returns 1 if the argument is prime, a 0 otherwise. Write a C program that invokes this function to generate prime numbers between the given range.

14.2 PLAN OF SOLUTION

- A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself.
- To check whether a given number is prime or not we divide the number starting from 2 till n/2. If it is completely divisible then the number is said to be not prime else the number is prime.

14.3 ALGORITHM

Input: Two numbers (n1 and n2) between which prime numbers are to be generated.

Output: To display the result after rotation.

Step 1: **Start**

Step 2: Read an integer **n** to test for primality

Step 3: Check **if n is less than or equal to zero**. If true **goto Step 4** otherwise **goto Step 5**

Step 4: **Display zero or negative integers by definition cannot be prime and goto Step13**

Step 5: Compute:

res = prime (n)

Step 6: Check **if res is equal to one**. It true **goto Step 7** otherwise **goto Step 8**

Step 7: **Display n is PRIME**

Step 8: **Display n is NOT PRIME**

Step 9: Read the range between which the prime numbers has to be generated: **n1 and n2**

Step 10: Initialize:

i = n1 + 1

Step 11: Check **if i is less than n2**. If true **goto Step 12** otherwise **goto Step 16**

Step 12: Compute:

range = prime (i)

Step 13: Check **if range is equal to one**. It true **goto Step 14** otherwise **goto Step 15**

Step 14: Compute:

Display i

Step 15: Increment i **goto Step 11**

Step 16: Stop

INT PRIME (INT N)

Step 1: Initialize:

i = 2

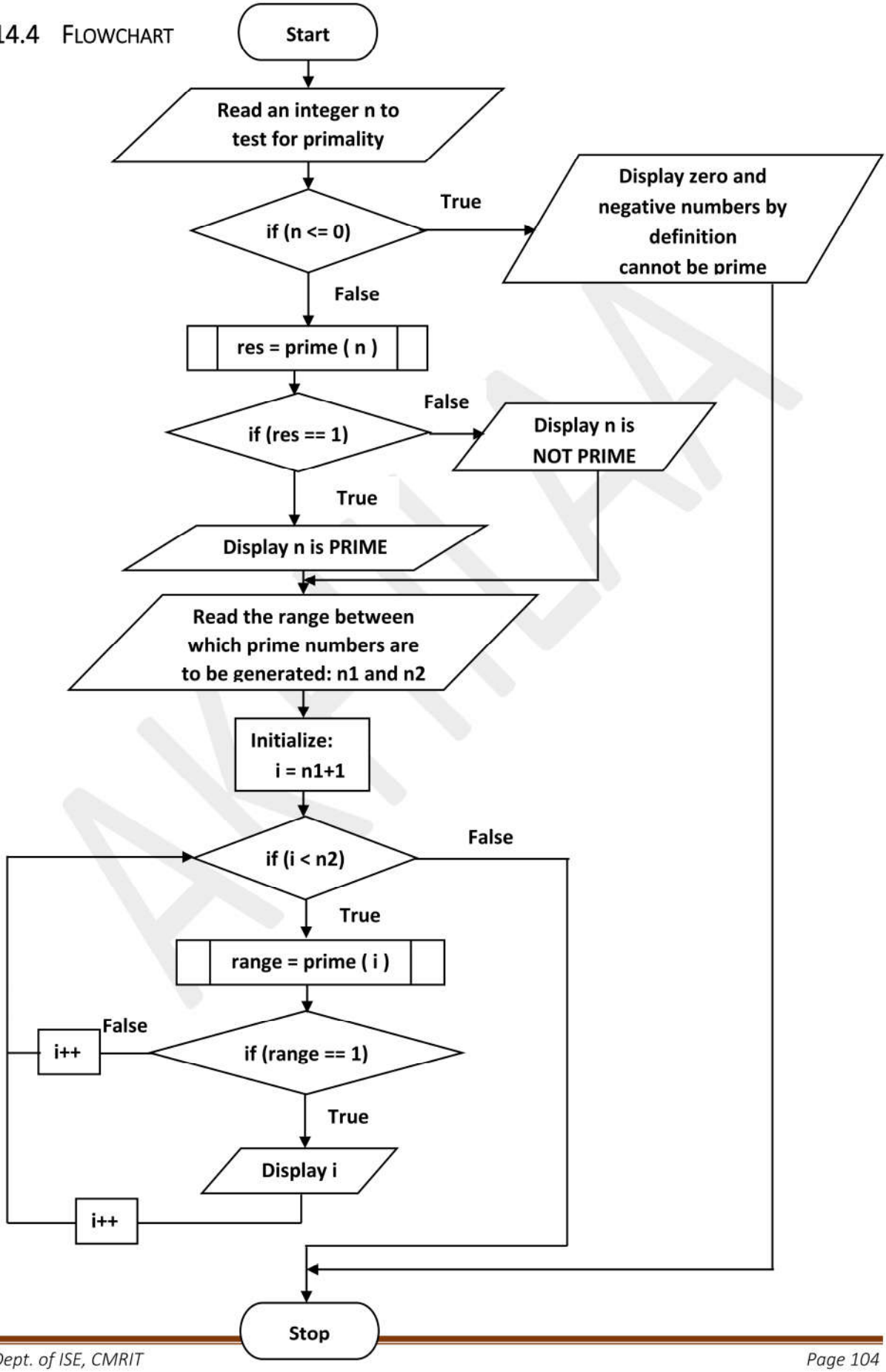
Step 2: Check **if i is less than or equal to n / 2**. If true **goto Step 3** otherwise **goto Step 5**

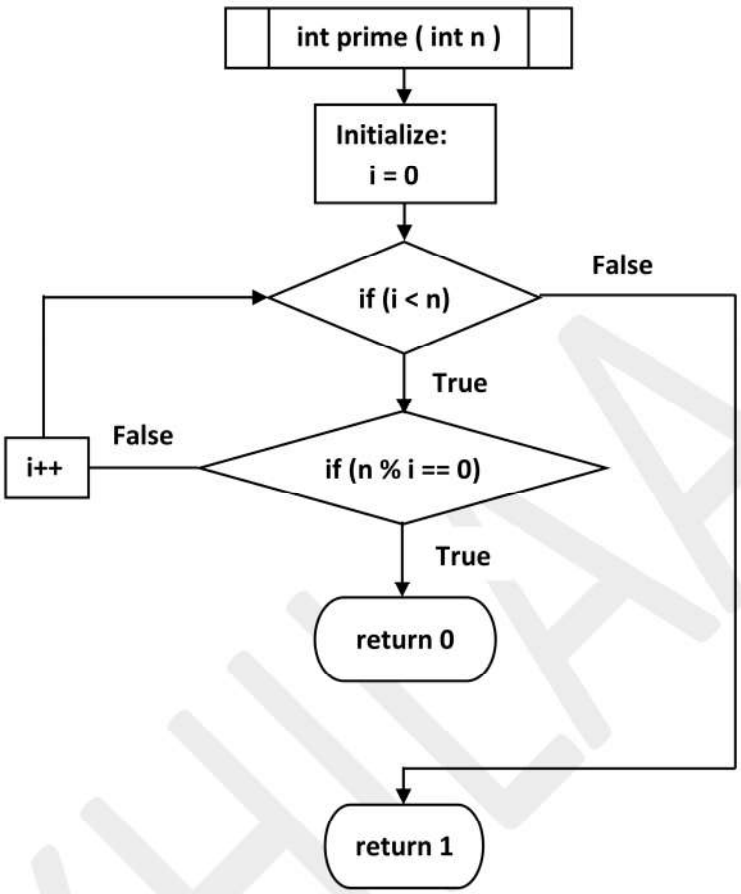
Step 3: Check **if n % i is equal to zero**. If true **return zero, break** otherwise **goto Step 4**

Step 4: Increment i **goto Step 2**

Step 5: Return one

14.4 FLOWCHART





14.5 PROGRAM CODE

```
/* Program to generate prime numbers between a given range. */

#include<stdio.h>

// Function declaration.
int prime ( int n );

int main()
{
        int n, n1, n2, i, res, range;

        // Input a number that you want to check for prime.
        printf ( "\nEnter a positive integer to test for primality: " );
        scanf ( "%d", &n );

        // Check if the entered number is positive or not.
        if ( n <= 0 )
        {
```

```
                printf ( "\nError: zero or -ve integers by definition cannot be
                prime!\n" );
                return 1;
        }
        // Fuction call.
        res = prime ( n );

        // Check whether the entered number is prime or not.
        if ( res == 1 )
                printf ( "\n%d is PRIME\n", n );
        else
                printf ( "\n%d is NOT PRIME\n", n );

        // Input the range to generate prime numbers.
        printf("\nEnter two numbers between which the prime numbers has to be
        generated: ");
        scanf("%d%d", &n1, &n2);

        // Display prime numbers between the specified range.
        printf("\nPrime numbers between %d and %d are :\n", n1, n2);
        for ( i = n1+1; i < n2; i++)
        {
                range = prime ( i );
                if ( range == 1 )
                        printf("%d\t",i);
        }
        printf("\n");

        return 0;
}

// Function definition.
int prime ( int n )
{
        int i;

        for ( i = 2 ; i <= n/2 ; i++ )
        {
                if ( n % i == 0 )
                {
                        return 0;       // if i divides n then NOT Prime number.
                }
        }

        return 1;       // Prime number.
}
```

14.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit prime.c
akhilaa@akhilaa-VirtualBox:~$ cc prime.c
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter a positive integer to test for primality: 5

5 is PRIME

Enter two numbers between which the prime numbers has to be generated: 2 50

Prime numbers between 2 and 50 are :
3       5       7       11      13      17      19      23      29      31      37      41
43      47
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter a positive integer to test for primality: 6

6 is NOT PRIME

Enter two numbers between which the prime numbers has to be generated: 23 73

Prime numbers between 23 and 73 are :
29      31      37      41      43      47      53      59      61      67      71

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter a positive integer to test for primality: 1036

1036 is NOT PRIME

Enter two numbers between which the prime numbers has to be generated: 203 253

Prime numbers between 203 and 253 are :
211     223     227     229     233     239     241     251

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter a positive integer to test for primality: 5051

5051 is PRIME

Enter two numbers between which the prime numbers has to be generated: 999
1050

Prime numbers between 999 and 1050 are :
```

```
1009   1013   1019   1021   1031   1033   1039   1049

akhilaa@akhilaa-VirtualBox:~$
```

14.7 ALTERNATE PROGRAM CODES

➢ **/* Program to generate prime numbers between a given range. */**

```
#include<stdio.h>

// Function declaration.
int prime ( int n );

int main()
{
        int n, n1, n2, i, res, range;

        // Input a number that you want to check for prime.
        printf ( "\nEnter a positive integer to test for primality: " );
        scanf ( "%d", &n );

        // Check if the entered number is positive or not.
        if ( n <= 0 )
        {
                printf ( "\nError: zero or -ve integers by definition cannot be
                prime!\n" );
                return 1;
        }

        // Fuction call.
        res = prime ( n );

        // Check whether the entered number is prime or not.
        if ( res == 1 )
                printf ( "\n%d is PRIME\n", n );
        else
                printf ( "\n%d is NOT PRIME\n", n );

        // Input the range to generate prime numbers.
        printf("\nEnter two numbers between which the prime numbers has
        to be generated: ");
        scanf("%d%d", &n1, &n2);

        // Display prime numbers between the specified range.
        printf("\nPrime numbers between %d and %d are :\n", n1, n2);
        for ( i = n1+1; i < n2; i++)
```

```
        {
                range = prime ( i );
                if ( range == 1 )
                        printf("%d\t",i);
        }
        printf("\n");

        return 0;
}

// Function definition.
int prime ( int n )
{
        int i;

        for ( i = 2 ; i < n ; i++ )
        {
                if ( n % i == 0 )
                {
                        return 0;       // if i divides n then NOT Prime
                                        // number.
                }
        }

        return 1;       // Prime number.
}
```

- **/* Program to generate prime numbers between a given range. */**

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

// Function declaration.
int prime ( int n );

int main()
{
        int n, n1, n2, i, res, range;

        // Input a number that you want to check for prime.
        printf ( "\nEnter a positive integer to test for primality: " );
        scanf ( "%d", &n );

        // Check if the entered number is positive or not.
        if ( n <= 0 )
        {
                printf ( "\nError: zero or -ve integers by definition cannot be
```

```
        prime!\n" );
        return 1;
}


// Fuction call.
res = prime ( n );

// Check whether the entered number is prime or not.
if ( res == 1 )
        printf ( "\n%d is PRIME\n", n );
else
        printf ( "\n%d is NOT PRIME\n", n );

// Input the range to generate prime numbers.
printf("\nEnter two numbers between which the prime numbers has
to be generated: ");
scanf("%d%d", &n1, &n2);

// Display prime numbers between the specified range.
printf("\nPrime numbers between %d and %d are :\n", n1, n2);
for ( i = n1+1; i < n2; i++)
{
        range = prime ( i );
        if ( range == 1 )
                printf("%d\t",i);
}
printf("\n");

return 0;
}

// Function definition.
int prime ( int n )
{
        int i;

        for ( i = 2 ; i <= sqrt ( n ) ; i++ )
        {
                if ( n % i == 0 )
                {
                        return 0;       // if i divides n then NOT Prime
                                        // number.
                }
        }

        return 1;       // Prime number.
```

```
}
```

15 BINOMIAL COEFFICIENT (VTU-PROGRAM 11)

15.1 PROBLEM STATEMENT

Draw the flowchart and write a recursive C function to find the factorial of a number, n!, defined by fact (n) = 1, if n = 0. Otherwise fact (n) = n * fact (n – 1). Using this function, write a C program to compute binomial coefficient ${}_nC_r$. Tabulate the results for different values of n and r with suitable messages.

15.2 PLAN OF SOLUTION

The factorial of a non-negative integer n, denoted by n!, is the product of all positive integers less than or equal to n. The value of 0! is 1

For example,

$$5! = 5 \text{ X } 4 \text{ X } 3 \text{ X } 2 \text{ X } 1$$

PERMUTATION:

An arrangement is called a Permutation. It is the rearrangement of objects or symbols into distinguishable sequences. When we set things in order, we say we have made an arrangement. When we change the order, we say we have changed the arrangement. So each of the arrangement that can be made by taking some or all of a number of things is known as Permutation.

$$^nP_r = n! / (n - r)!$$

COMBINATION:

A Combination is a selection of some or all of a number of different objects. It is an un-ordered collection of unique sizes .In a permutation the order of occurrence of the objects or the arrangement is important but in combination the order of

occurrence of the objects is not important. A binomial coefficient equals the number of combinations of *r* items that can be selected from a set of *n* items.

$$^{n}C_{r} = {}^{n}P_{r} / r!$$

where n, r are non negative integers and r <= n. r is the size of each permutation. n is the size of the set from which elements are permuted. !is the factorial operator.

15.3 ALGORITHM

Input: Input two positive numbers n and r.

Output: To compute binomial coefficient $_{n}C_{r}$.

Step 1: Start

Step 2: Read the values of **n** and **r**

Step 3: Check **if n is less than or equal to r**. If true **goto Step 4** otherwise **goto Step 5**

Step 4: Display the value of n should be greater than r and goto Step 8

Step 5: Check **if n is less than zero or r is less than zero**. If true **goto Step 6** otherwise **goto Step 7**

Step 6: Display the value of n or r should not be negative and goto step 8

Step 7: Compute:

factorial = fact (n)

Display factorial

ncr = fact (n) / (fact (n – r) * fact (r))

Display ncr

Step 8: Stop

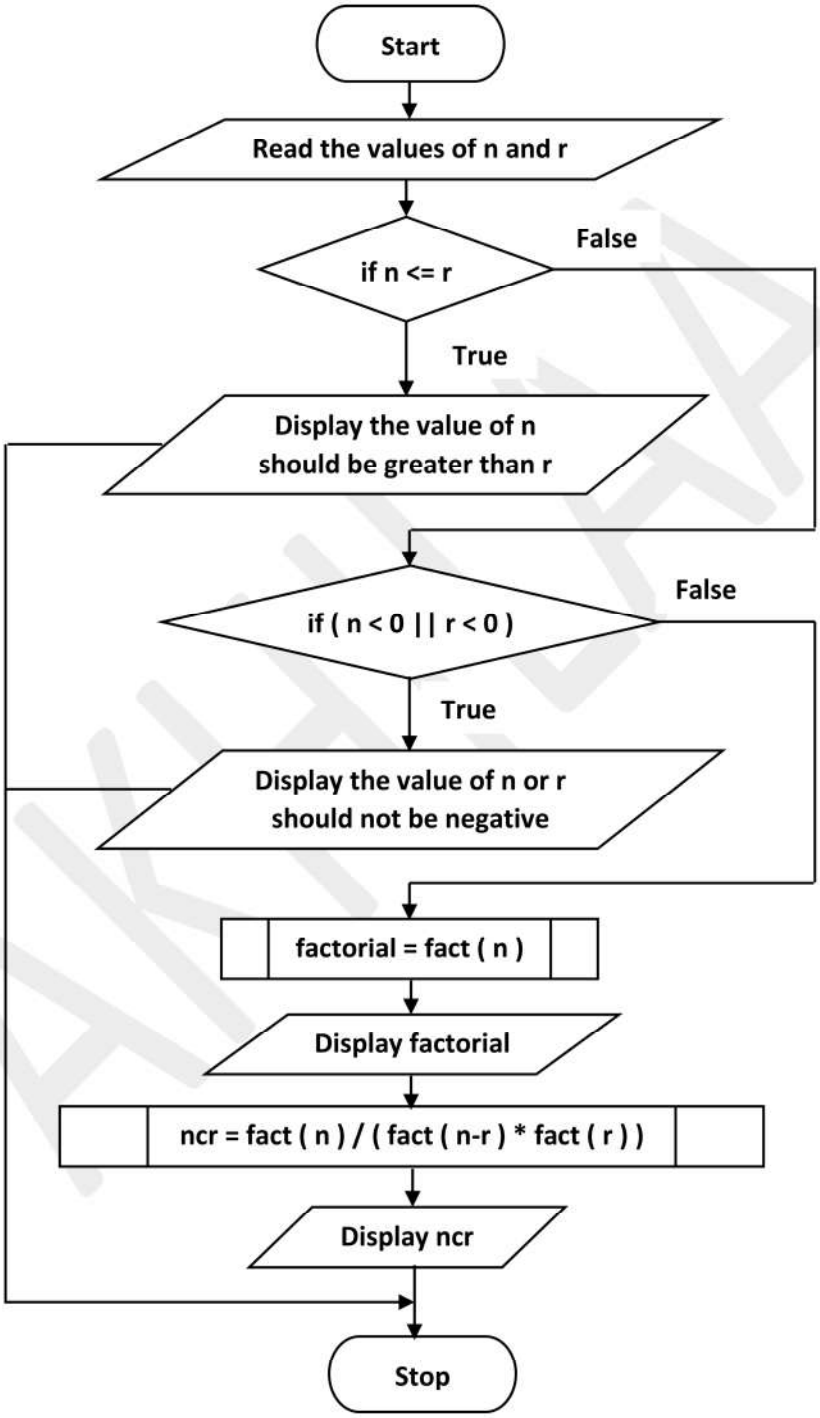
LONG INT FACT (LONG INT N)

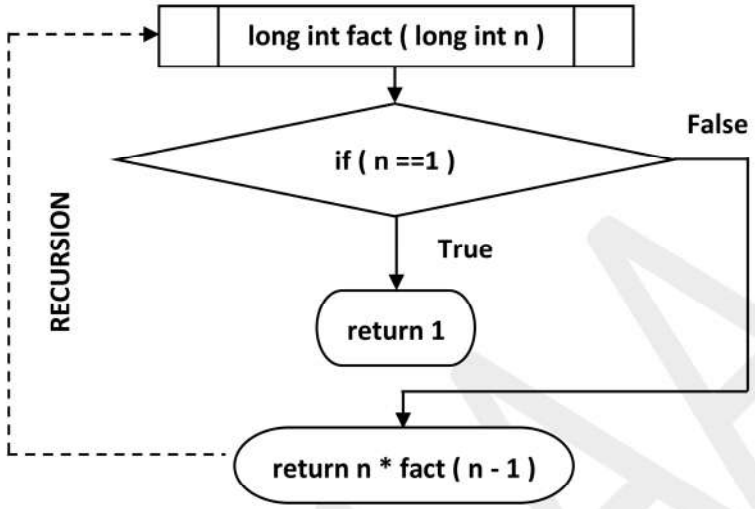
Step 1: Check **if n is equal to one**. If true **goto Step 2** otherwise **goto Step 3**

Step 2: Return one

Step 3: Return the value n * find_fact (n – 1)

15.4 FLOWCHART





15.5 PROGRAM CODE

/* Program to compute Binomial Coefficient (nCr = n! / ((n-r)! * r!)). */

```
#include<stdio.h>

// Function declaration.
long int fact ( long int n );

int main()
{
        long int n, r, factorial;
        double ncr, npr;

        // Input the values of n and r.
        printf ( "\nEnter the values of n and r: " );
        scanf ( "%ld%ld", &n, &r );

        if ( n <= r )
        {
                printf ( "\nError!!! Entered value of n should be greater than r\n\n" );
                return 0;
        }
        else if ( n < 0 || r < 0 )
        {
                printf ( "\nError!!! n & r should not be -ve\n\n" );
                return 0;
        }
```

```
        else
        {
                // Function call.
                factorial = fact ( n );
                printf ( "\nFactorial of %ld is: %ld\n", n, factorial );
                // To compute binomial coefficient. ( n! / ( ( n - r )! * r! ) )
                ncr = fact ( n ) / ( fact ( n – r ) * fact ( r ) );
                printf ( "\n%ld C %ld = %.0lf\n\n", n, r, ncr );
        }

        return 0;
}

// Function definition to compute fatorial. ( n! )
long int fact ( long int n )
{
        if ( n == 0 || n == 1 )
                return 1;

        return n * fact ( n - 1 );
}
```

15.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit binomial_coefficient.c
akhilaa@akhilaa-VirtualBox:~$ cc binomial_coefficient.c
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the values for n and r: 6 4

Factorial of 6 is: 720

The binomial coefficient 6 C 4 = 15

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the values for n and r: 10 5

Factorial of 10 is: 3628800

The binomial coefficient 10 C 5 = 252

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the values for n and r: 5 2

Factorial of 5 is: 120
```

```
The binomial coefficient 5 C 2 = 10

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the values for n and r: 5 5

Error:: Entered value of n should be greater than r
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the values for n and r: 4 5

Error:: Entered value of n should be greater than r
akhilaa@akhilaa-VirtualBox:~$
```

15.7 VIVA QUESTIONS

- What is recursion?

16 UNIVERSITY FILES (VTU-PROGRAM 12)

16.1 PROBLEM STATEMENT

Given two university information files "studentname.txt" and "usn.txt" that contains students Name and USN respectively. Write a C program to create a new file called "output.txt" and copy the content of files "studentname.txt" and "usn.txt" into output file in the sequence shown below. Display the contents of output file "output.txt" on the screen.

Student Name	USN
Name 1	USN 1
Name 2	USN 2
.....	.....
.....	.....

Heading

16.2 PLAN OF SOLUTION

- A file represents a sequence of bytes on the disk where a group of related data is stored.
- To access a file, we need a file pointer that keeps track of the file being accessed.
- Here our aim is to copy the contents of sname.txt and usn.txt into out.txt file.
- We scan/take the contents from each of the file string wise and copy it into another file until end of file is reached.

16.3 ALGORITHM

Input: Two files usn.txt and sname.txt.

Output: To copy the contents of both the files into out.txt.

Step 1: Start

Step 2: Compute:

fp_usn = fopen ("usn.txt", "r")

Step 3: Check **if fp_usn is equal to NULL**. If true **goto Step 4** otherwise **goto Step 5**

Step 4: Display Error in opening file usn.txt

Step 5: Compute:

fp_sname = fopen ("sname.txt", "r")

Step 6: Check **if fp_sname is equal to NULL**. If true **goto Step 7** otherwise **goto Step 8**

Step 7: Display Error in opening file studentname.txt

Step 8: Compute:

fp_out = fopen ("out.txt", "w")

Step 9: Check **if fp_out is equal to NULL**. If true **goto Step 10** otherwise **goto Step 11**

Step 10: Display Error in opening file out.txt

Step 11: Repeat the below infinite times

Step 12: Compute:

fscanf (fp_sname, "%s", name)

fscanf (fp_usn, "%s", usn)

Step 13: Check **if both the files have not reached end of file**. If true **goto Step 14** otherwise **goto Step 15**

Step 14: Compute:

fprintf (fp_out, "\n%s\t%s", usn, name) **goto Step 11**

Step 15: Break from the loop

Step 16: Compute:

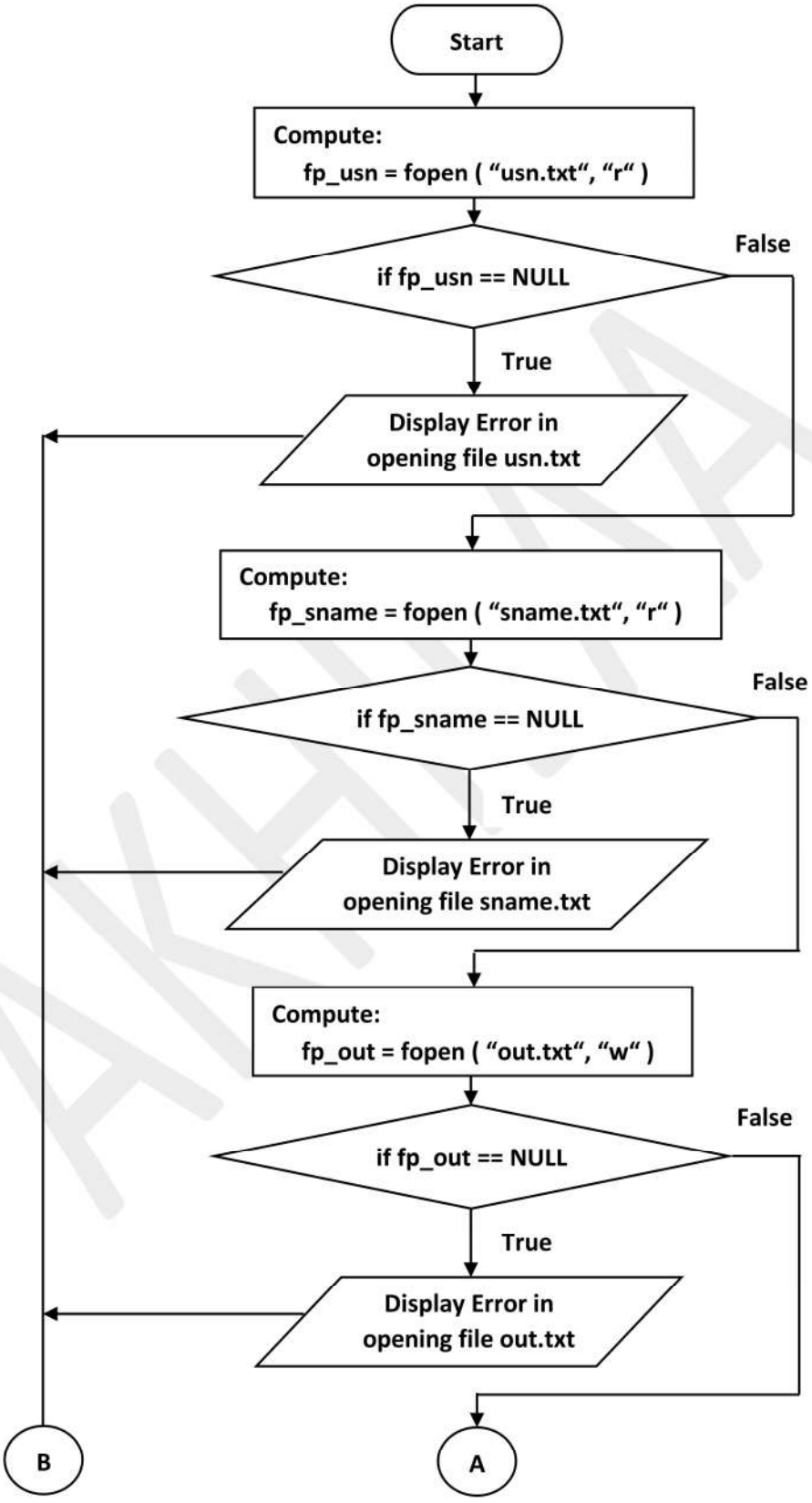
fclose (fp_usn)

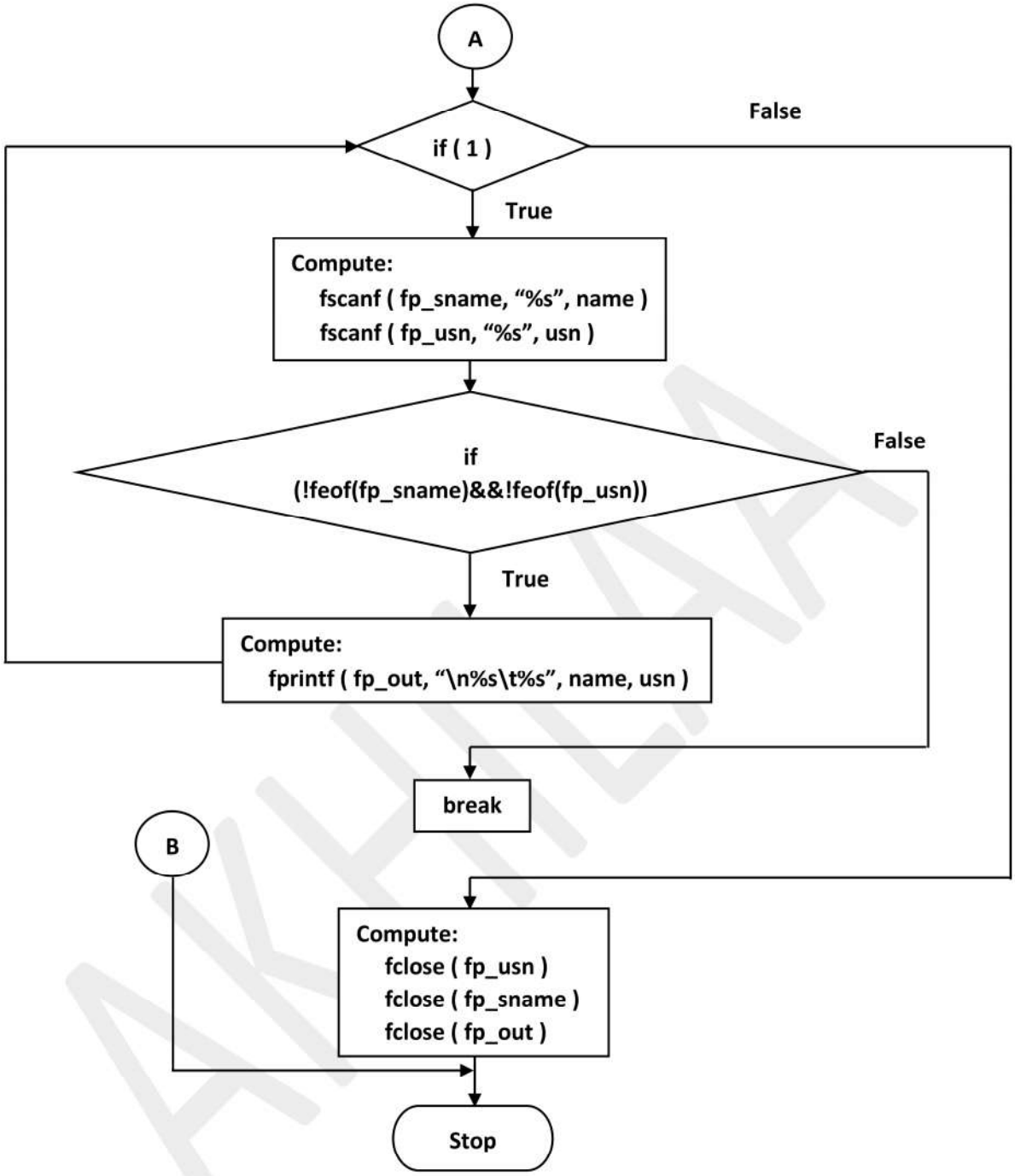
fclose (fp_sname)

fclose (fp_out)

Step 17: Stop

16.4 FLOWCHART





16.5 PROGRAM CODE

/* Program to merge contents of two files into an output file. */

```
#include<stdio.h>

int main()
{
        FILE *fp_usn, *fp_sname, *fp_out;
        int buff;
```

```
        // Create a usn.txt file for reading USN.
        fp_usn = fopen ( "usn.txt", "r" );
        if ( fp_usn == NULL || fp_sname == NULL || fp_out == NULL )
        {
                printf ( "Error in opening file 'usn.txt'\n");
                exit ( 0 );
        }

        // Create a sname.txt file for reading Student Name.
        fp_sname = fopen ( "sname.txt", "r" );
        if ( fp_sname == NULL )
        {
                printf ( "Error in opening file 'sname.txt'\n");
                exit ( 0 );
        }

        // Writing into out.txt file.
        fp_out = fopen ( "out.txt", "w" );
        if ( fp_out == NULL )
        {
                printf ( "Error in opening file 'out.txt'\n");
                exit ( 0 );
        }

        fprintf ( fp_out, "Student Name \t USN" );

        while ( 1 )
        {
                // Take/Scan the contents of sname.txt and usn.txt.
                fscanf ( fp_sname, "%s", name);
                fscanf ( fp_usn, "%s", usn );

                if ( !feof ( fp_sname ) && !feof ( fp_usn ) )
                        // copy to out.txt file.
                        fprintf ( fp_out, "\n%s\t%s", usn, name );
                else
                        break;
        }
        fclose ( fp_usn );
        fclose ( fp_sname );
        fclose ( fp_out );

        return 0;
}
```

16.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit university_files.c
akhilaa@akhilaa-VirtualBox:~$ cc university_files.c
akhilaa@akhilaa-VirtualBox:~$ ./a.out
akhilaa@akhilaa-VirtualBox:~$ cat out.txt
USN     Student Name
0001    sachin
0002    virat
0003    Dhoni
0004    ABD
0005    Ricky
akhilaa@akhilaa-VirtualBox:~$
```

16.7 ALTERNATE PROGRAM CODE

- **/* Program to merge contents of two files into an output file. */**

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

int main()
{
        FILE *fp_usn, *fp_sname, *fp_out;
        int buff;

        // Create a usn.txt file for reading USN.
        fp_usn = fopen ( "usn.txt", "r" );
        // Create a sname.txt file for reading Student Name.
        fp_sname = fopen ( "sname.txt", "r" );
        // Writing into out.txt file.
        fp_out = fopen ( "out.txt", "w" );

        if ( fp_usn == NULL || fp_sname == NULL || fp_out == NULL)
        {
                printf ( "Error in opening file 'usn.txt'\n");
                exit (0);
        }

        do
        {
                // Copy the contents from usn.txt to out.txt untill new line and end
                // of file is encountered.
                buff = fgetc ( fp_usn );
                if ( buff != '\n' && buff != EOF )
                        fputc ( buff, fp_out );
```

```
                // After encountering new line.
                if ( buff == '\n' )
                {
                        // Add one space.
                        fputc ( ' ', fp_out );
                        do
                        {
                                // Copy the contents from student.txt to out.txt untill
                                // end of file is encountered.
                                buff = fgetc ( fp_sname );
                                if ( buff != EOF )
                                        fputc ( buff, fp_out );

                                // When new line is encountered break the second
                                // loop.
                                if ( buff == '\n' )
                                        break;
                        } while ( 1 );
                }

        } while ( buff != EOF );

        fclose ( fp_usn );
        fclose ( fp_sname );
        fclose ( fp_out );

        return 0;
}
```

- **/* Program to merge contents of two files into an output file. */**

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

int main()
{
        FILE *fp_usn, *fp_sname, *fp_out;
        char usn [ 20 ], name [ 20];

        // Create a usn.txt file for reading USN.
        fp_usn = fopen ( "usn.txt", "r" );
        // Create a sname.txt file for reading Student Name.
        fp_sname = fopen ( "sname.txt", "r" );
        // Writing into out.txt file.
        fp_out = fopen ( "out.txt", "w" );

        if ( fp_usn == NULL || fp_sname == NULL || fp_out == NULL)
```

```
        {
                printf ( "Error in opening file\n");
                exit (0);
        }

        fprintf ( fp_out, "USN\tStudent Name\n" );
        fscanf ( fp_usn, "%s", usn );
        fscanf ( fp_sname, "%s", name );
        while ( !feof ( fp_usn ) && !feof ( fp_sname ) )
        {
                fprintf ( fp_out, "%s\t%s\n", usn, name );
                fscanf ( fp_usn, "%s", usn );
                fscanf ( fp_sname, "%s", name );
        }

        fclose ( fp_usn );
        fclose ( fp_sname );
        fclose ( fp_out );

        return 0;
}
```

16.8 VIVA QUESTIONS

- What is a file?
- How to create a file?
- How to create and open a file?
- What are fopen() and fclose()?
- What are fgetc() and fputc()?
- What are fgets() and fgets()?
- What are fprintf() and fscanf()?

17 STUDENT STRUCTURE (VTU-PROGRAM 13)

17.1 PROBLEM STATEMENT

Write a C program to maintain a record of n student details using an array of structures with four fields (Roll number, Name, Marks, and Grade). Assume appropriate data type for each field. Print the marks of the student, given the student name as input.

17.2 PLAN OF SOLUTION

- Create structure of type student with the following fields:

  rollno: int, name: char, marks: int and grade: char.
- For 'n' number of students input the student details.
- Given a particular student name, we have to display the marks of that student if record found otherwise display student record not found.
- To search for a given student name we adopt a searching technique called Linear Search.
- Linear Search is a method for finding a particular value in a list that checks each element in sequence until the desired element is found or the list is exhausted.

17.3 ALGORITHM

Input: Student details such as rollno, name, marks and grade.

Output: To search for a particular student details given the name.

Define a structure of type student with rollno, name, marks, grade as fields.

Step 1: Start

Step 2: Read the number of students: **num_studs**

Step 3: Initialize:

i = 0

Step 4: Check if i is less than num_studs. If true **goto Step 5** otherwise **goto Step 6**

Step 5: Compute:

Read the following details for student i:

Read Roll No: **stud_arr [i].rollno**

Read Name: **stud_arr [i].name**

Read Marks: **stud_arr [i].marks**

Read Grade: **stud_arr [i].grade** **goto Step 4**

Step 6: Read the student name to be searched: **stud_name**

Step 7: Initialize:

i = 0

Step 8: Check **if i is less than num_studs**. If true **goto Step 9** otherwise **goto Step 11**

Step 9: Check **if stud_name and stud_arr[i].name are equal to zero**. If true **goto Step 10** otherwise **goto Step 11**

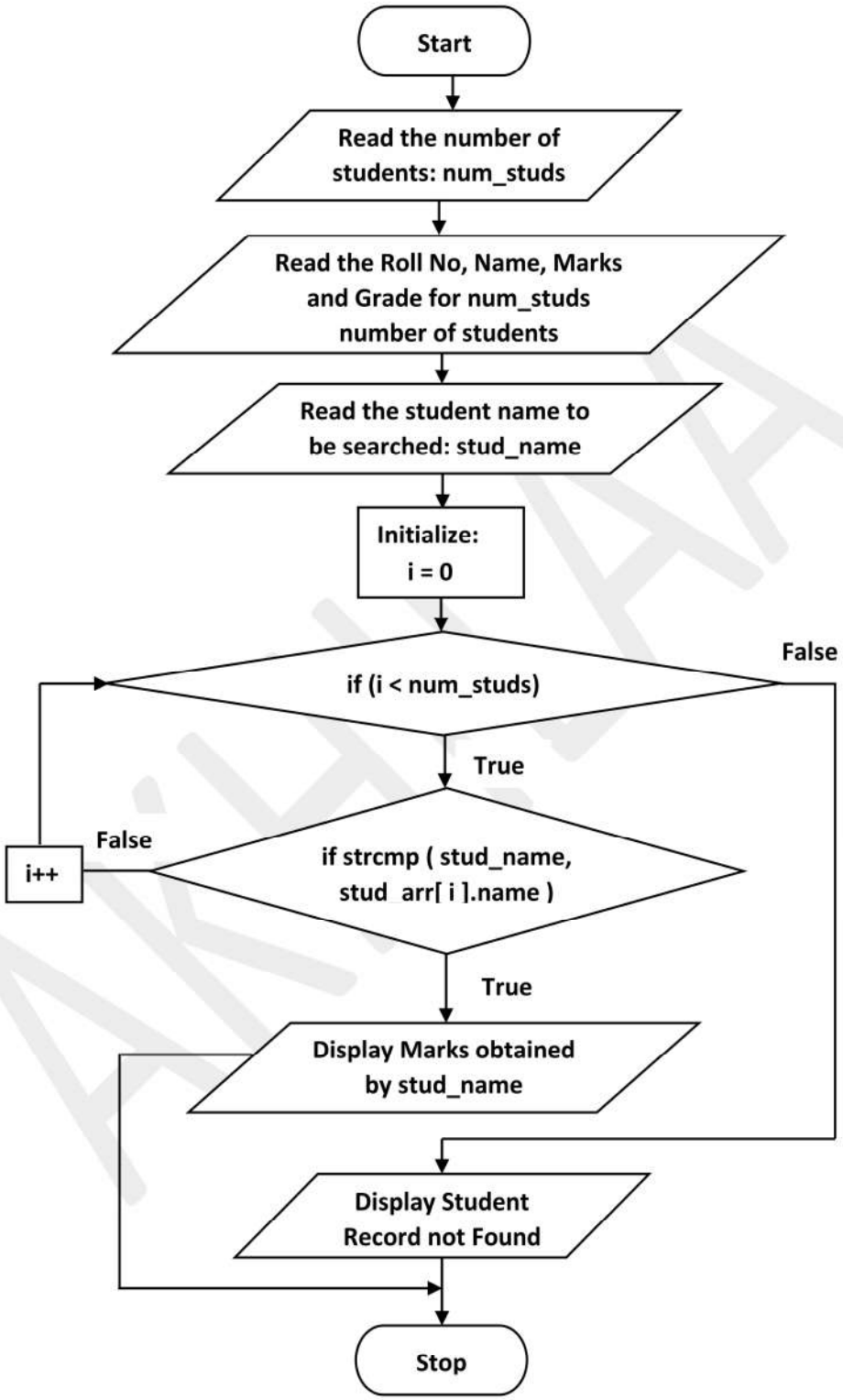
Step 10: Display Marks obtained by Student and goto Step 13

Step 11: Increment i

Step 12: Display Student Record not Found

Step 13: Stop

17.4 FLOWCHART



17.5 PROGRAM CODE

```
/* Program to create details of a student using structures. */

#include<stdio.h>
#include<string.h>

// Name of defined structure type is student.
struct student
{
        int rollno;
        char name [ 30 ];
        int marks;
        char grade [ 2 ];
};
struct student stud_arr [ 30 ];

int main()
{
        int i, num_studs;
        char stud_name [ 30 ];

        // Input the number of students.
        printf ( "\nEnter the number of students: " );
        scanf ( "%d" , &num_studs );

        // Input student details.
        for ( i = 0 ; i < num_studs ; i++ )
        {
                printf ( "\n\nEnter the following details for Student %d", i );

                printf ( "\nRoll No: " );
                scanf ( "%d" , &stud_arr[i].rollno );

                printf ( "\nName: " );
                scanf ( "%s" , stud_arr[i].name );

                printf ( "\nMarks: " );
                scanf ( "%d" , &stud_arr[i].marks );

                printf ( "\nGrade: " );
                scanf ( "%s" , stud_arr[i].grade );
        }

        // Input the student name that you want to search for.
        printf ( "\n\nEnter the student name you wish to search: " );
        scanf ( "%s" , stud_name );
```

```
        // Linear Search.
        for ( i = 0 ; i < num_studs ; i++ )
        {
                if ( strcmp ( stud_name, stud_arr[i].name ) == 0 )
                {
                        printf ( "\nMarks obtained by %s is: %d\n\n",
                        stud_arr[i].name, stud_arr[i].marks );
                        return 0;
                }
        }

        // Display record not found.
        printf ( "\nStudent Record not Found!!!!\n\n" );

        return 0;
}
```

17.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit student_details.c
akhilaa@akhilaa-VirtualBox:~$ cc student_details.c
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number of students: 3

Enter the following details for Student 0
Roll No: 1

Name: sachin

Marks: 100

Grade: A

Enter the following details for Student 1
Roll No: 2

Name: virat

Marks: 99

Grade: A

Enter the following details for Student 2
Roll No: 3
```

Name: ABD

Marks: 98

Grade: A

Enter the student name you wish to search: virat

Marks obtained by virat is: 99

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the number of students: 3

Enter the following details for Student 0
Roll No: 1

Name: sachin

Marks: 100

Grade: A

Enter the following details for Student 1
Roll No: 2

Name: virat

Marks: 99

Grade: A

Enter the following details for Student 2
Roll No: 3

Name: ABD

Marks: 98

Grade: A

Enter the student name you wish to search: dhoni

Student Record not Found!!!!

akhilaa@akhilaa-VirtualBox:~$

17.7 ALTERNATE PROGRAM CODE

```
/* Program to create details of a student using structures. */

#include<stdio.h>
#include<string.h>

// Name of defined structure type is student.
typedef struct
{
        int rollno;
        char name [ 30 ];
        int marks;
        char grade;
}student;

int main()
{
        student stud_arr [ 30 ];        // Since student is typedef, we dont need the
                                        // struct keyword.
        int i, num_studs;
        char stud_name [ 30 ];

        // Input the number of students.
        printf ( "\nEnter the number of students: " );
        scanf ( "%d" , &num_studs );

        // Input student details.
        for ( i = 0 ; i < num_studs ; i++ )
        {
                printf ( "\n\nEnter the following details for Student %d", i );

                printf ( "\nRoll No: " );
                scanf ( "%d" , &stud_arr[i].rollno );

                printf ( "\nName: " );
                scanf ( "%s" , stud_arr[i].name );

                printf ( "\nMarks: " );
                scanf ( "%d" , &stud_arr[i].marks );

                printf ( "\nGrade: " );
                scanf ( "%s" , &stud_arr[i].grade );
        }

        // Input the student name that you want to search for.
        printf ( "\n\nEnter the student name you wish to search: " );
```

```
        scanf ( "%s" , stud_name );

        // Linear Search.
        for ( i = 0 ; i < num_studs ; i++ )
        {
                if ( strcmp ( stud_name, stud_arr[i].name ) == 0 )
                {
                        printf ( "\nMarks obtained by %s is: %d\n\n",
                        stud_arr[i].name, stud_arr[i].marks );
                        return 0;
                }
        }

        printf ( "\nStudent Record not Found!!!!\n\n" );

        return 0;
}
```

17.8 VIVA QUESTIONS

- What is a structure?
- What is the difference between arrays and structures?
- What is the syntax for creating a structure?
- What is the size of a structure?
- How to declare a structure variable?
- How to access a structure variable?
- What is typedef?

18 SUM MEAN AND STANDARD DEVIATION (VTU-PROGRAM 14)

18.1 PROBLEM STATEMENT

Write a C program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of n real numbers.

18.2 PLAN OF SOLUTION

$$Sum = \sum_{i=1}^{n} x_i$$

$$Mean = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$Standard\ Deviation = \sigma = \sqrt{\sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}}$$

18.3 ALGORITHM

Input: An array of n numbers.

Output: To compute sum, mean, and standard deviation.

Step 1: Start

Step 2: Read the value of **n**

Step 3: Read the numbers: **a [n]**

Step 4: Compute:

sum = find_sum (a, n)

Display the sum is: sum

mean = sum / n

Display the mean is: mean

sd = find_sd (a, mean, n)

Display the standard deviation: sd

Step 6: Stop

DOUBLE FIND_SUM (DOUBLE *A, INT N)

Step 1: Initialize:

sum = 0

i = 0

Step 2: Check **if i is less than n**. If true **goto Step 3** otherwise **goto Step 4**

Step 3: Compute:

sum = sum + * (a + i)

Increment i **goto Step 2**

Step 4: Return the value of sum

DOUBLE FIND_SD (DOUBLE *A, INT N)

Step 1: Initialize:

total = 0

i = 0

Step 2: Check **if i is less than n**. If true **goto Step 3** otherwise **goto Step 4**

Step 3: Compute:

temp = *(a + i) - mean

temp = temp * temp

total = total + temp

Increment i **goto Step 2**

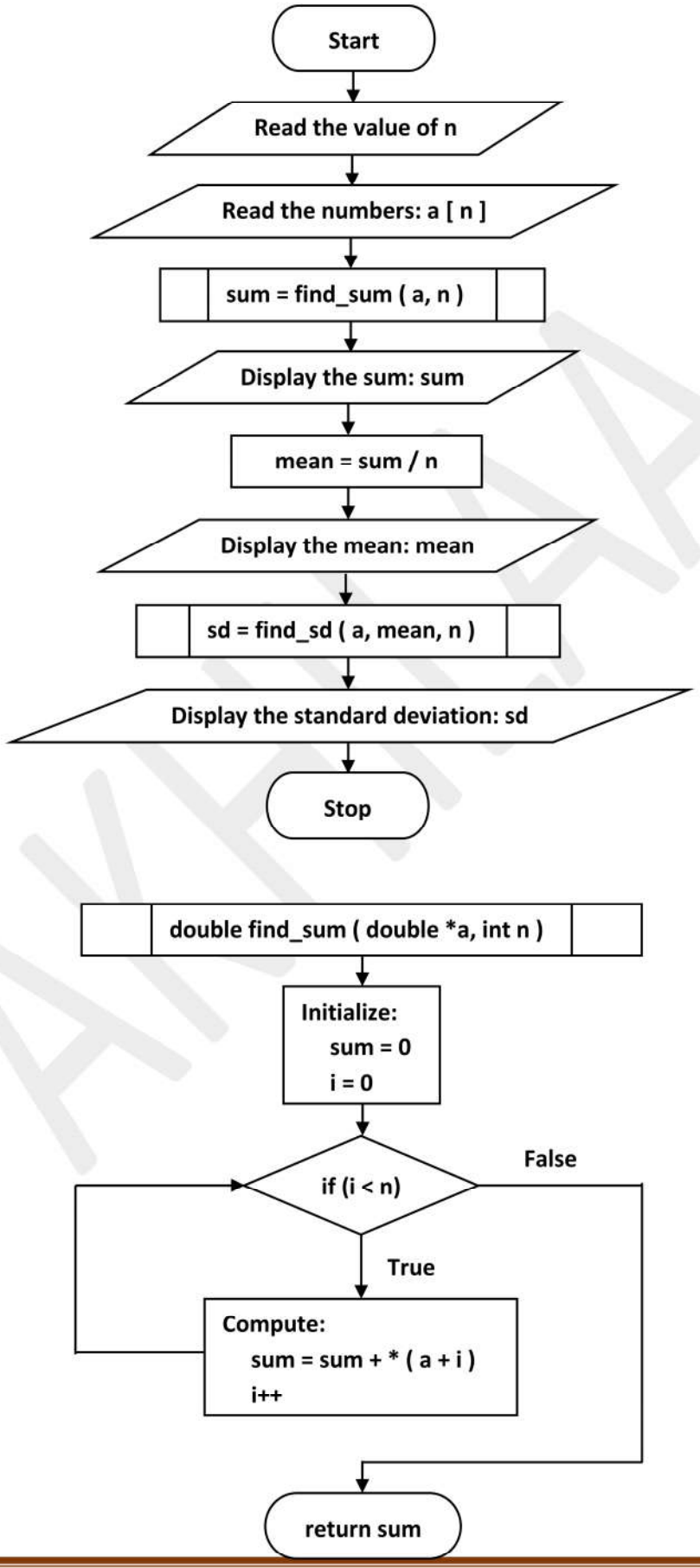
Step 4: Compute:

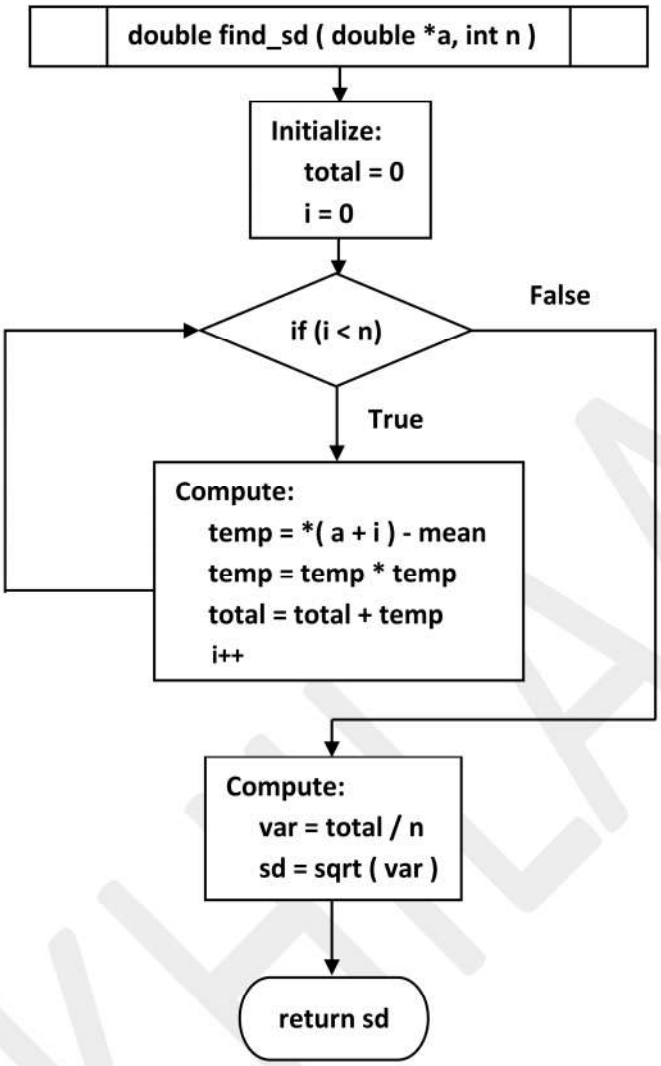
var = total / n

sd = sqrt (var)

Return the value of sd

18.4 FLOWCHART





18.5 PROGRAM CODE

/* Program to find sum, mean and standard deviation using pointers. */

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

// Function declarations.
double find_sum ( double *a, int n );
double find_sd ( double *a, double mean, int n );

int main()
{
        int n, i;
        double a [ 10 ], sum, mean, sd;
```

```
        // Read the number of integers.
        printf ( "\nEnter the value of n: " );
        scanf ( "%d", &n );

        // Read the integers.
        printf ( "\nEnter %d numbers: ", n );
        for ( i = 0; i < n ; i++ )
        {
                scanf ( "%lf", &a [ i ] );
        }

        // Function call for find_sum.
        sum = find_sum ( a, n );
        printf ( "\nThe Sum is: %lf\n", sum );

        mean = sum / n;
        printf ( "\nThe Mean is: %lf\n", mean );

        // Function call for find_sd.
        sd = find_sd ( a, mean, n );
        printf ( "\nThe Standard Deviation is: %lf\n", sd );

        return 0;
}


// Function definition to compute sum of the given numbers.
double find_sum ( double *a, int n )
{
        int i;
        double sum = 0;

        for ( i = 0; i < n ; i++ )
        {
                sum = sum + *( a + i );
        }

        return sum;
}

// Function definition to compute mean.
double find_mean ( double *a, int n )
{
        double mean;

        mean = find_sum ( a, n ) / n;
```

```
        return mean;
}

// Function definition to compute standard deviation.
double find_sd ( double *a, double mean, int n )
{
        int i;
        double temp, total = 0, var, sd;

        for ( i = 0 ; i < n ; i++ )
        {
                temp = *( a + i ) - mean;
                temp = temp * temp;
                total = total + temp;
        }

        var = total / n;
        sd = sqrt ( var );

        return sd;
}
```

18.6 SAMPLE RUNS

```
akhilaa@akhilaa-VirtualBox:~$ gedit sum_mean_sd.c
akhilaa@akhilaa-VirtualBox:~$ cc sum_mean_sd.c -lm
akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the value of n: 5

Enter 5 numbers:
1
2
3
4
5

The Sum is: 15.000000

The Mean is: 3.000000

The Standard Deviation is: 1.414214

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the value of n: 6

Enter 6 numbers:
```

```
14.6
5.5
3.8
26.4
19.7
2.1

The Sum is: 72.100000

The Mean is: 12.016667

The Standard Deviation is: 8.952731

akhilaa@akhilaa-VirtualBox:~$ ./a.out

Enter the value of n: 3

Enter 3 numbers:
55.5
88.8
22.2

The Sum is: 166.500000

The Mean is: 55.500000

The Standard Deviation is: 27.189336

akhilaa@akhilaa-VirtualBox:~$
```

18.7 ALTERNATE PROGRAM CODES

- **/* Program to find sum, mean and standard deviation using pointers. */**

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

int main()
{
        int n, i;
        double a [ 10 ], sum, mean, sd, total, var;

        // Read the number of integers.
        printf ( "\nEnter the value of n: " );
        scanf ( "%d", &n );

        // Read the integers.
```

```
        printf ( "\nEnter %d numbers: ", n );
        for ( i = 0; i < n ; i++ )
        {
                scanf ( "%lf", (a+i) );
        }

        //Compute sum.
        sum = 0;
        for ( i = 0 ; i < n ; i++ )
        {
                sum = sum + *(a+i);
        }
        printf ( "\nThe Sum is: %lf\n", sum );

        // Compute mean.
        mean = sum / n;
        printf ( "\nThe Mean is: %lf\n", mean );

        // Compute variance and standard deviation.
        total= 0;
        for ( i = 0; i < n ; i++)
        {
                total = total + pow ( (*(a+i)-mean), 2);
        }
        var = total / n;
        sd = sqrt ( var );
        printf ( "\nThe Standard Deviation is: %lf\n\n", sd );

        return 0;
}
```

- **/* Program to find sum, mean and standard deviation using pointers. */**

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

int main()
{
        int n, i;
        double a [ 10 ], *p, sum, mean, sd, total, var;

        // Read the number of integers.
        printf ( "\nEnter the value of n: " );
        scanf ( "%d", &n );

        // Read the integers.
        printf ( "\nEnter %d numbers: ", n );
```

```
        for ( i = 0; i < n ; i++ )
        {
                scanf ( "%lf", (a+i) );
        }

        //Compute sum.
        p = a;
        sum = 0;
        for ( i = 0 ; i < n ; i++ )
        {
                sum = sum + *p;
                *p++;
        }
        printf ( "\nThe Sum is: %lf\n", sum );

        // Compute mean.
        mean = sum / n;
        printf ( "\nThe Mean is: %lf\n", mean );

        // Compute variance and standard deviation.
        p = a;
        total= 0;
        for ( i = 0; i < n ; i++)
        {
                total = total + pow ( (*p-mean), 2);
                *p++;
        }
        var = total / n;
        sd = sqrt ( var );
        printf ( "\nThe Standard Deviation is: %lf\n\n", sd );

        return 0;
}
```

18.8 VIVA QUESTIONS

- What is a pointer?
- How to declare a pointer variable?
- What is dangling pointer?
- What is pointer to a pointer?
- What is pointer to an array?
- What is pass by reference?