

C LAB OBSERVATION NOTE

LAB 1

Aim- Learn to use basic arithmetic operators.

Implement C programs for the following problem statements:

a. Input two numbers and compute all arithmetic operations.

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

```
int main()
```

```
{
```

```
int a = 9,b = 4, c;
```

```
c = a+b;
```

```
printf("a+b = %d \n",c);
```

```
c = a-b;
```

```
printf("a-b = %d \n",c);
```

```
c = a*b;
```

```
printf("a*b = %d \n",c);
```

```
c = a/b;
```

```
printf("a/b = %d \n",c);
```

```
c = a%b;
```

```
printf("Remainder when a divided by b = %d \n",c);

return 0;
}
```

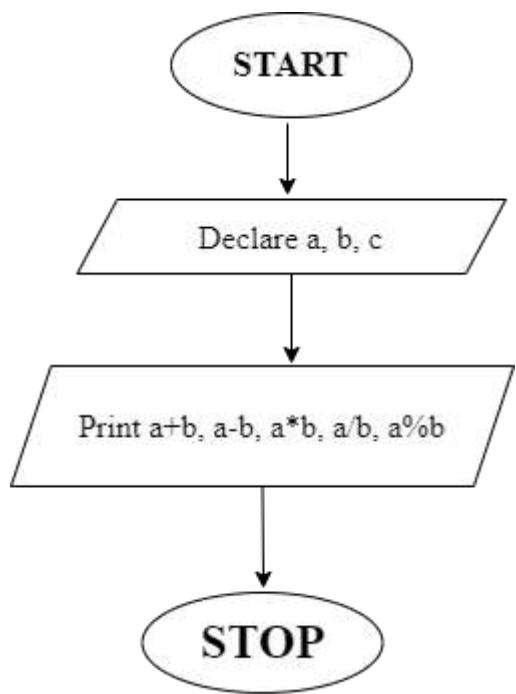
```
main.c
1 #include <stdio.h>
2 int main()
3 {
4     int a = 10, b = 3;
5
6     printf("a+b = %d\n", a+b);
7     printf("a-b = %d\n", a-b);
8     printf("a*b = %d\n", a*b);
9     printf("a/b = %d\n", a/b);
10    printf("Remainder when a divided by b = %d \n", a%b);
11
12    return 0;
13 }
```

```
a+b = 13
a-b = 5
a*b = 36
a/b = 2
Remainder when a divided by b = 1

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- Value of arithmetic operators get printed variables a and b are operated upon by respective operators '+', '-','*', '/ ',and '%'

a) 1. START
 2. Read a, b
 3. $c \leftarrow a + b$
 4. Print c
 5. $c \leftarrow a - b$
 6. Print c
 7. $c \leftarrow a * b$
 8. Print c
 9. $c \leftarrow a / b$
 10. Print c
 11. $c \leftarrow a \% b$
 12. Print c
 13. STOP



b. Input radius, compute area, diameter, & circumference of the circle

```
#include <stdio.h>

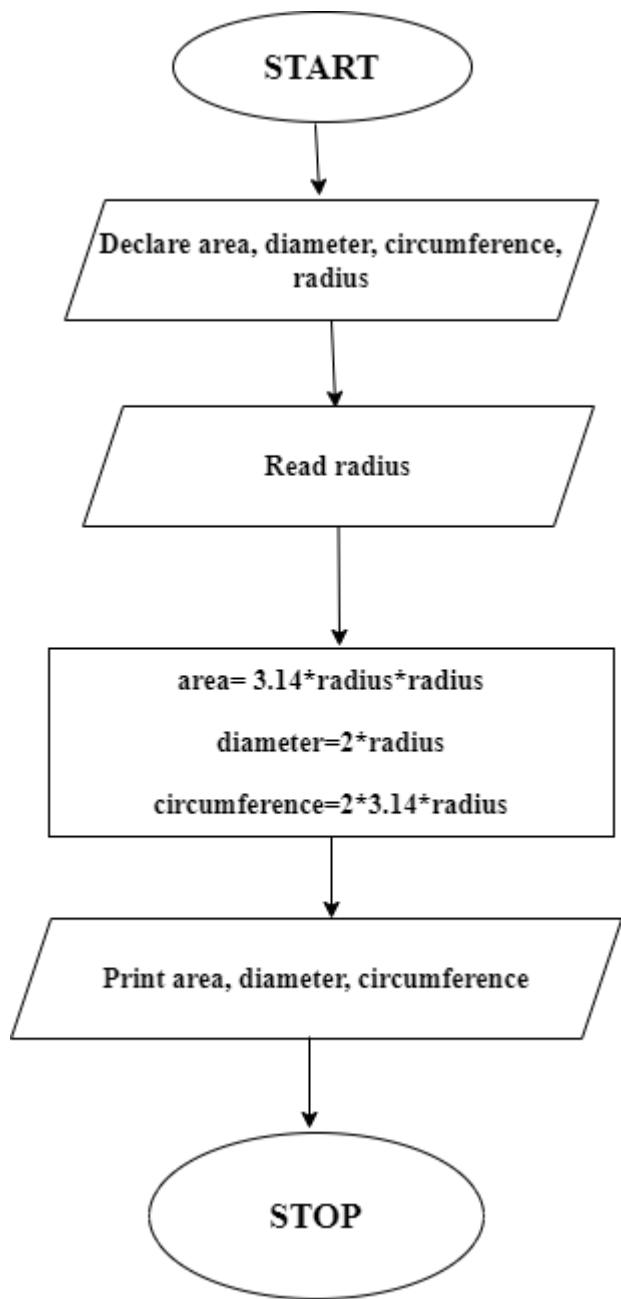
int main()
{
    float area, diameter, circumference;
    int radius;
    printf ("please enter radius=");
    scanf ("%d", &radius);
    area = 3.14*radius*radius;
    printf ("area=%f\n", area);
    diameter= 2*radius;
    printf ("diameter=%f\n", diameter);
    circumference= 2*radius*3.14;
    printf ("circumference=%f\n", circumference);
    return 0;
}
```

```
main.c
1 #include <stdio.h>
2 int main()
3 {
4     float pi=3.14,radius,diameter,area,circumference;
5     printf("please enter radius=");
6     scanf("%f",&radius);
7     area=pi*radius*radius;
8     diameter=2*radius;
9     circumference=2*pi*radius;
10    printf("area=%f\n",area);
11    printf("diameter=%f\n",diameter);
12    printf("circumference=%f\n",circumference);
13 }
please enter radius=5
area=78.500000
diameter=10.000000
circumference=31.400000

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- Value of area, diameter, and circumference gets printed after we enter value of radius in output and press enter key.

b) 1. START
2. Read radius
3. area = $3.14 \times \text{radius} \times \text{radius}$
4. Print area
5. diameter = $2 \times \text{radius}$
6. Print diameter
7. circumference = $2 \times \text{radius} \times 3.14$
8. Print circumference
9. STOP



c. Swapping the values of two variables using third variable.

```
#include<stdio.h>

int main() {
    double var1, var2, temp;
    printf("Enter first number: ");
    scanf("%lf", &var1);
```

Samuela Abigail Mathew
71762108039

```

printf("Enter second number: ");
scanf("%lf", &var2);
temp = var1;
var1 = var2;
var2 = temp;
printf("\nAfter swapping, first number = %.2lf\n", var1);
printf("After swapping, second number = %.2lf", var2);
return 0;
}

```

```

main.c
1 #include<stdio.h>
2 int main() {
3     double var1, var2, temp;
4     printf("Enter first number: ");
5     scanf("%lf", &var1);
6     printf("Enter second number: ");
7     scanf("%lf", &var2);
8     temp = var1;
9     var1 = var2;
10    var2 = temp;
11    printf("\nAfter swapping, first number = %.2lf\n", var1);
12    printf("After swapping, second number = %.2lf", var2);
13    return 0;
14 }

Enter first number: 4
Enter second number: 5

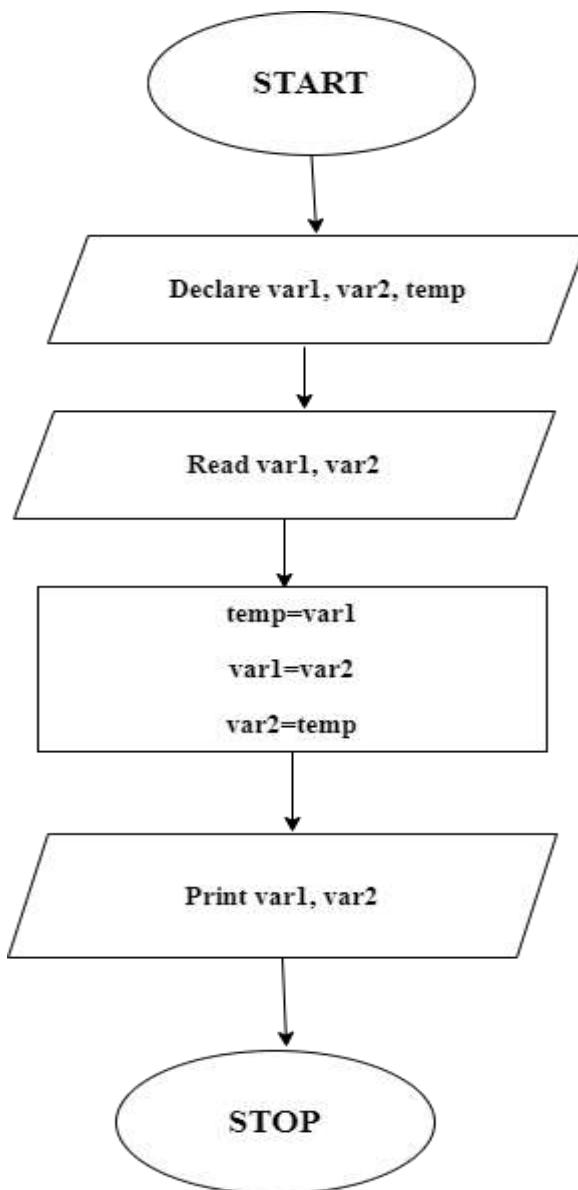
After swapping, first number = 5.00
After swapping, second number = 4.00

...Program finished with exit code 0
Press ENTER to exit console.

```

Inference- Numbers get swapped with the help of 3rd variable ‘temp’ after typing value of variables.

c) 1. START
 2. Read var1, var2
 3. temp \leftarrow var1
 4. var1 \leftarrow var2
 5. var2 \leftarrow temp
 6. Print var1, var2
 7. STOP



d.Program for swapping the values of two variables without using a third variable.

```
#include <stdio.h>

int main()
{
    double var1, var2;
    printf("Enter first number: ");
    scanf("%lf", &var1);
    printf("Enter second number: ");
    scanf("%lf", &var2);

    var1 = var1 + var2;
    var2 = var1 - var2;
    var1 = var1 - var2;
    printf("\nAfter swapping, first number = %.2lf\n", var1);
    printf("After swapping, second number = %.2lf", var2);

    return 0;
}
```

The screenshot shows a web-based C compiler interface. At the top, there are buttons for Run, Debug, Stop, Share, Save, and Back. The code editor window contains a file named 'main.c' with the following content:

```
1 #include <stdio.h>
2 int main()
3 {
4     double var1, var2;
```

Below the code editor is a terminal window displaying the program's output:

```
Enter first number: 8
Enter second number: 56

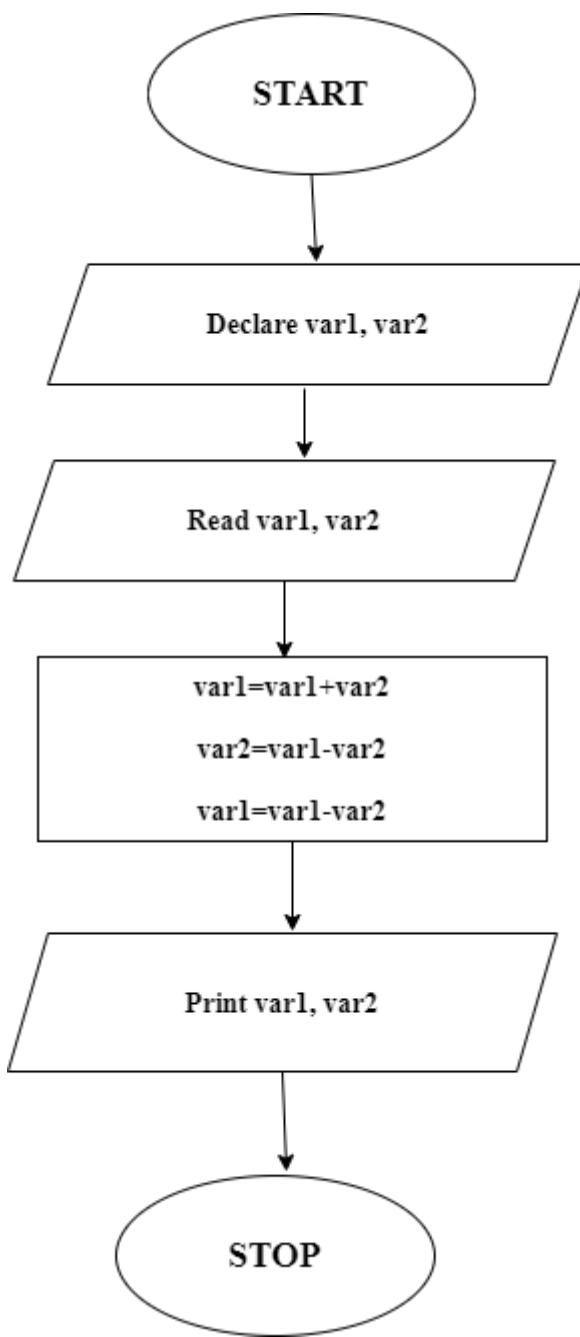
After swapping, first number = 56.00
After swapping, second number = 8.00

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- Numbers get swapped without use of 3rd variable by using '+' and '-' operators instead.

Handwritten notes for a program to swap two variables:

- d) 1. START
2. Read var1, var2
3. var1 \leftarrow var1 + var2
4. var2 \leftarrow var1 - var2
5. var1 \leftarrow var1 - var2
6. Print var1, var2
7. STOP



e. To evaluate algebraic expression $(ax+b)/(ax-b)$.

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

```
int main()
```

```
{
```

```
int a,b,x;  
float k;  
printf ("enter value of a,b,x=");  
scanf ("%d%d%d",&a,&b,&x);
```

```
k= (a*x+b)/(a*x-b);  
printf ("k=%f", k);
```

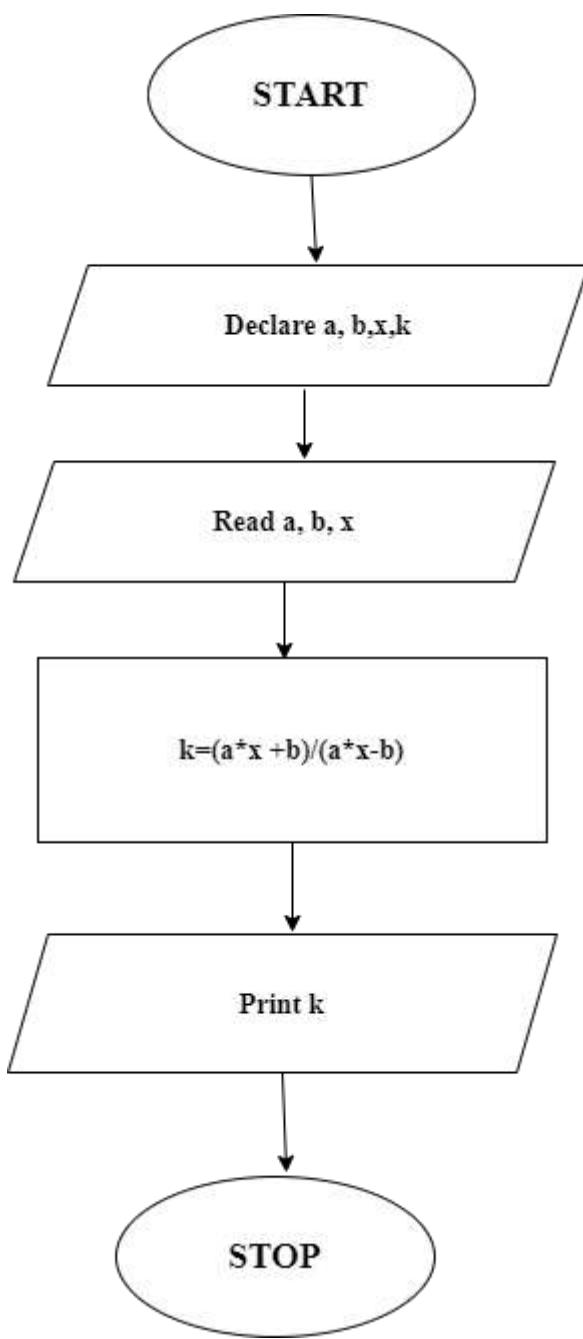
```
return 0;
```

```
}
```

```
Run | Debug | Stop | Share | Save  
main.c  
1 #include <stdio.h>  
2 int main()  
3 {  
4     int a,b,x;  
5     float k;  
6     printf ("enter value of a,b,x=");  
7     scanf ("%d%d%d",&a,&b,&x);  
8  
9     k= (a*x+b)/(a*x-b);  
10    printf ("k=%f", k);  
11  
12    return 0;  
13 }  
  
enter value of a,b,x=3 4 2  
k=5.000000  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Inference- We get value of given algebraic expression after typing value of variables in output and pressing enter key.

e) 1. START
 2. Read a, b, x
 3. $k \leftarrow (a^*x + b) / (a^*x - b)$
 4. Print k
 5. STOP



LAB 1 -Take Home

Implement C programs for the following problem

statements:

a. Expression Evaluation: The prescribed expression has a single pair of parentheses missing that makes it syntactically wrong and it does not compile. In your c code snippet, place a pair of parentheses in proper places to correct the expression and evaluate it for x = 3, y =5,z =15, w = -1;

Given Expression: $x * w + y * z = 7$ (syntactically wrong)

Corrected Expression: $k= x * w + y *(z = 7)$

Value of the corrected Expression: 32

Inference- syntax errors make the codes unable to be compiled. Here, x=3,w=-1, y=5, and z=7. So output is $k= x * w + y *(z = 7)$ which is $k=-3+35$.So k=32

b.Expression Evaluation: What would be the value of the given expressions by assuming the following operand values?

Expression1: (int) $(c * y / z + y / z * c)$

Expression2: $x - s * t * - c - u$

Expression3: (float) $(x + y < z + w \&& a > b - 17 * x || !x < 5)$

```
int w = 0, x = 2.5, y = 5, z = 3, r, s = 4, t = 5, u = -3;  
double a = 2.36, b = 3.19, c = 3.0, d = 2.91726;  
  
#include <stdio.h>  
  
int main()  
{  
    int w = 0, x = 2.5, y = 5, z = 3, r, s = 4, t = 5, u = -3, m, n;  
    double a = 2.36, b = 3.19, c = 3.0, d = 2.91726;  
    float o;  
  
    m= (int) (c * y / z + y / z * c);  
    n= (x - s * t *- c - u);  
    o= (float) (x + y < z + w && a > b - 17 * x || !x < 5);  
  
    printf("Value of given expression m= %d\n", m);  
    printf("Value of given expression n= %d\n", n);  
    printf("Value of given expression o= %f\n", o);  
  
    return 0;  
}
```

The screenshot shows a C IDE interface with the following details:

- File Menu:** File, Open, Save, Run, Debug, Stop, Share.
- Code Editor:** The file "main.c" is open, containing the following code:

```
1 #include <stdio.h>
2 int main()
3 {
4     int m= 8, n= 65, o= 1.000000;
5 }
```
- Output Window:** Displays the program's output:

```
Value of given expression m= 8
Value of given expression n= 65
Value of given expression o= 1.000000

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- Parentheses is very important. Even a simple mistake can give a different answer. So correct output is m=8, n=65, and o=1.000000

Handwritten notes for a programming assignment:

- a) 1. START
2. Read w, x, y, z, s, t, u, a, b, c, d
3. $m \leftarrow c * y / z + y / z * c$
4. $n \leftarrow (x - 8 * t - c - u) *$
5. $o \leftarrow (x + y < x + w \&\& a > b - 17 * x)$
 $\{ !x < 5 \}$
6. Print m, n, o
7. STOP

c.Pattern Display: Implement a C code snippet to print the pattern below.

Sample input 1: 3

Sample input 2: 5

```
#include <stdio.h>

int main() {
    int rows,col,i,j;
    printf("Enter rows=");
    scanf ("%d", &rows);
    col=2*rows;
    //printf ("rows=%d col=%d\n", rows, col);

    for (i=1; i<=rows; i++)
    {
        //printf("i=%d",i);
        for (j=1; j<=col; j++)
        {
            if((j<=i) || (j>=(col-i+1)))
                printf ("*");
            else
                printf (" ");
        }
        printf("\n");
    }
}
```

Samuela Abigail Mathew
71762108039

```
return 0;  
}  
  
main.c
```

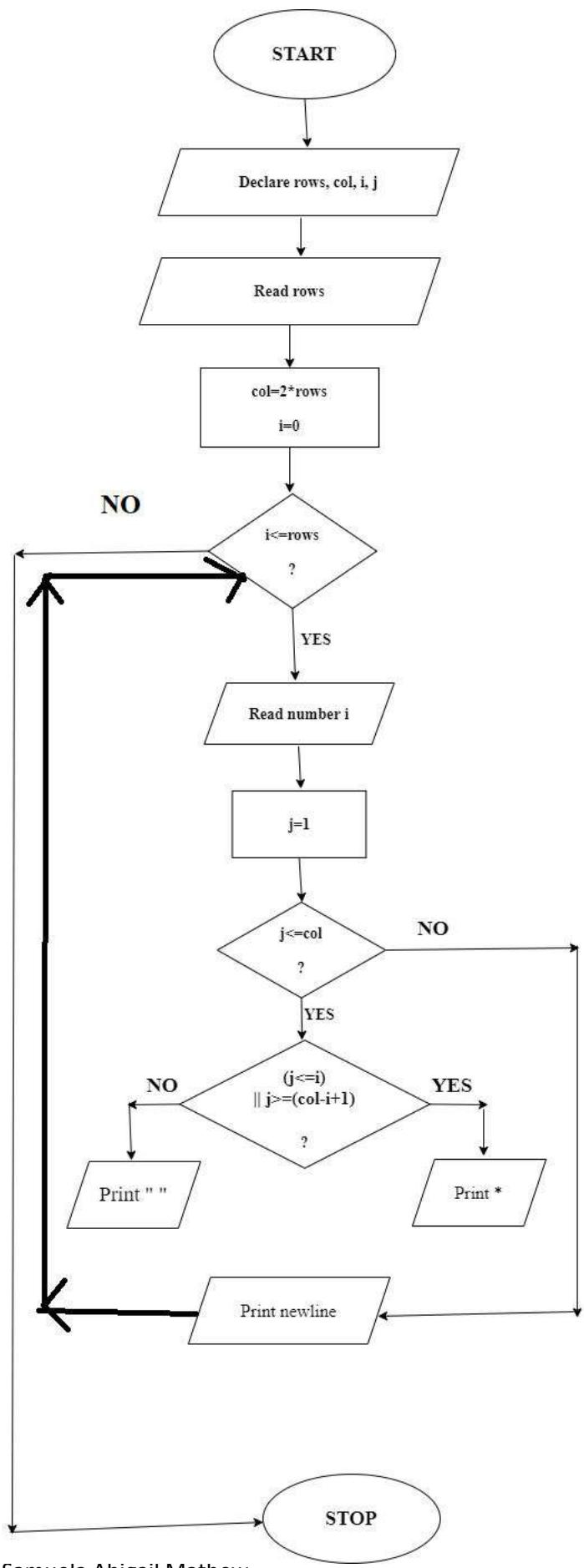
```
1 #include <stdio.h>  
2  
3 int main() {  
4     int i, j, rows, cols;
```

```
Enter rows=5  
rows=5 col=10  
*      *  
**      **  
***      ***  
****      ****  
*****      *****
```

```
...Program finished with exit code 0  
Press ENTER to exit console.
```

Inference- Pyramid made of stars is formed on entering value for number of rows in output. It is based on increment and decrement.

1. START
2. Read rows
3. col $\leftarrow 2 * \text{rows}$
4. for $i \leftarrow 1$ to rows do
5. for $j \leftarrow 1$ to cols do
6. if $j \leq 1$ OR $j \geq (\text{col} - i + 1)$
7. Print *
8. Else
9. Print " "
10. end for
11. Print "\n"
12. end for
13. STOP



d. Leap Year: A calendar year is said to be a leap year if it occurs four years once and has 29 days for the month of February. One of your friends Advaith was born on 29th of February. He wishes to know how many times till date he has celebrated his birthday (29th February). Take his birth year and present year as inputs and print the number of times he has experienced 29th of February. Note if present year is a leap year, do include its 29th of February as well. You may also display an error as “Birth Year is incorrect” if the birth year entered is not a leap year.

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

```
int main() {  
    int a, b, year=0;  
    printf("Please enter year of birth ");  
    scanf("%d",&a);  
    printf("Please enter current year ");  
    scanf("%d",&b);  
  
    if(((a%4 == 0) && (a%100!= 0)) || ((a%4 == 0) && (a%400 == 0)))  
        // Code for calculating number of leap years.  
    {  
        year = (b/4 - b/400 + b/100)-(a/4 - a/400 + a/100)+1;  
    }
```

```

printf ("No: of Leap years between %d and %d is %d", a,b,
year);
}

else
{
printf ("Entered year %d is not a leap year!", a);

}

return 0;
}

```

The screenshot shows an online C compiler interface. The code in the editor is:

```

main.c
1 printf("Please enter year of birth ", 
2 scanf("%d",&a);
3 printf("Please enter current year ");
4 scanf("%d",&b);

```

The terminal window displays the following output:

```

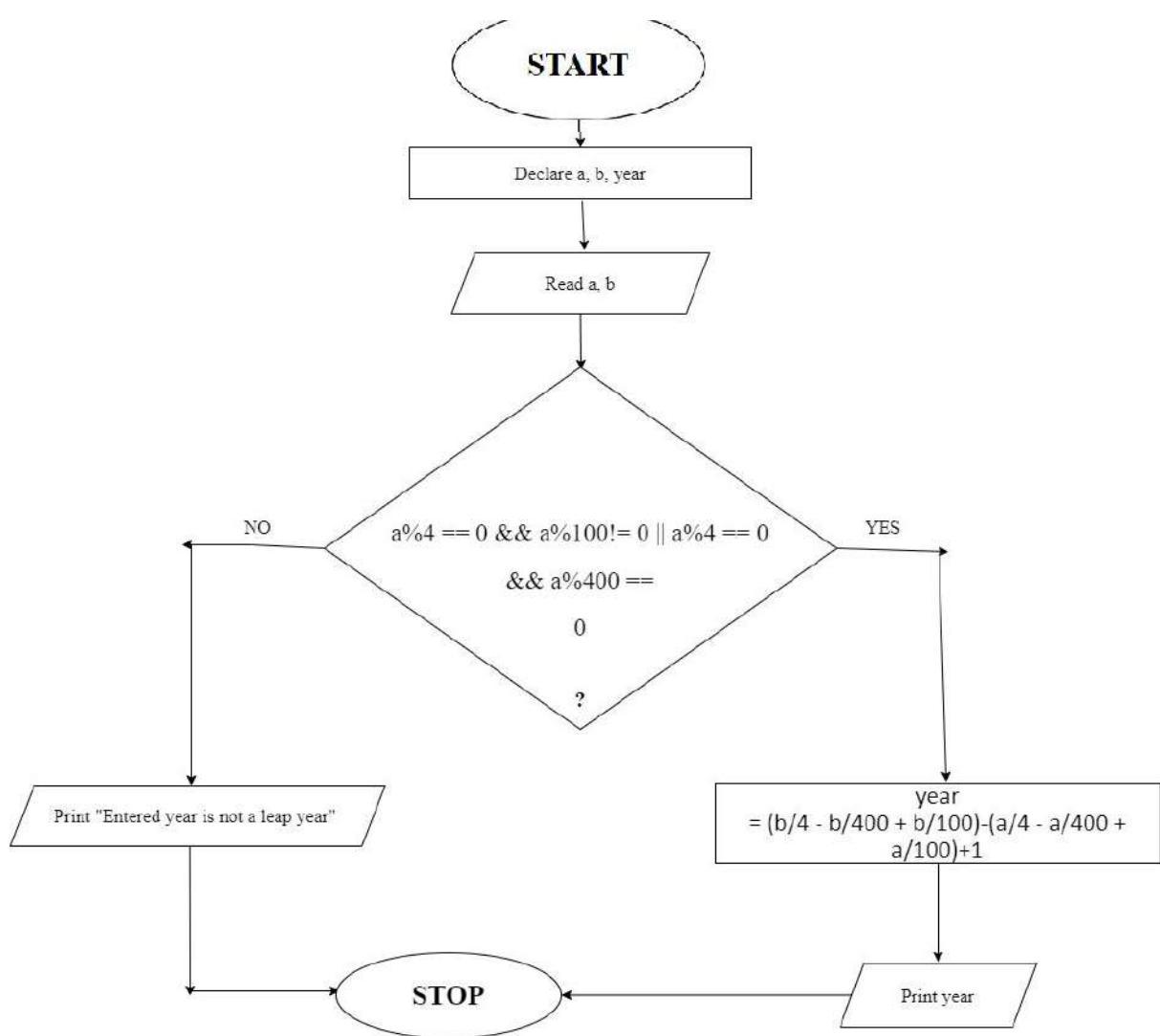
Please enter year of birth 2000
Please enter current year 2022
No: of Leap years between 2000 and 2022 is 6

...Program finished with exit code 0
Press ENTER to exit console. []

```

Inference- When calculating we should do it in such a way that the result includes year of birth also.

1. START
 2. Read a, b
 3. If $a \% 4 = 0$ AND $a \% 100 \neq 0$ OR
 $a \% 4 = 0$ AND $a \% 400 = 0$
 4. $year = (b/4 - b/400 + b/100) - (a/4 - a/400 + a/100) + 1$
 5. Print year
 6. Else
 7. Print "not a leap year!"
 8. STOP



LAB 2

Aim- Learn to use control statements (part 1).

Implement C programs for the following problem

statements:

a)To find sum of individual digits of a positive and negative integer.

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

```
int main()
```

```
{
```

```
int n,sum=0;
```

```
printf("enter integer=");
```

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

```
while (n != 0)
```

```
{
```

```
sum=sum+n%10;
```

```
n=n/10;
```

```
}
```

```
if (sum<0)
```

```
{
```

Samuela Abigail Mathew
71762108039

```

sum= (-1)*sum;

printf ("sum of individual digits of the given integer is
%d",sum);

}

else

{

printf("sum of individual digits of the given integer is
%d",sum);

}

return 0;

}

```

```

main.c
4 int n,sum=0;
5 printf("enter integer=");
6 scanf("%d",&n);

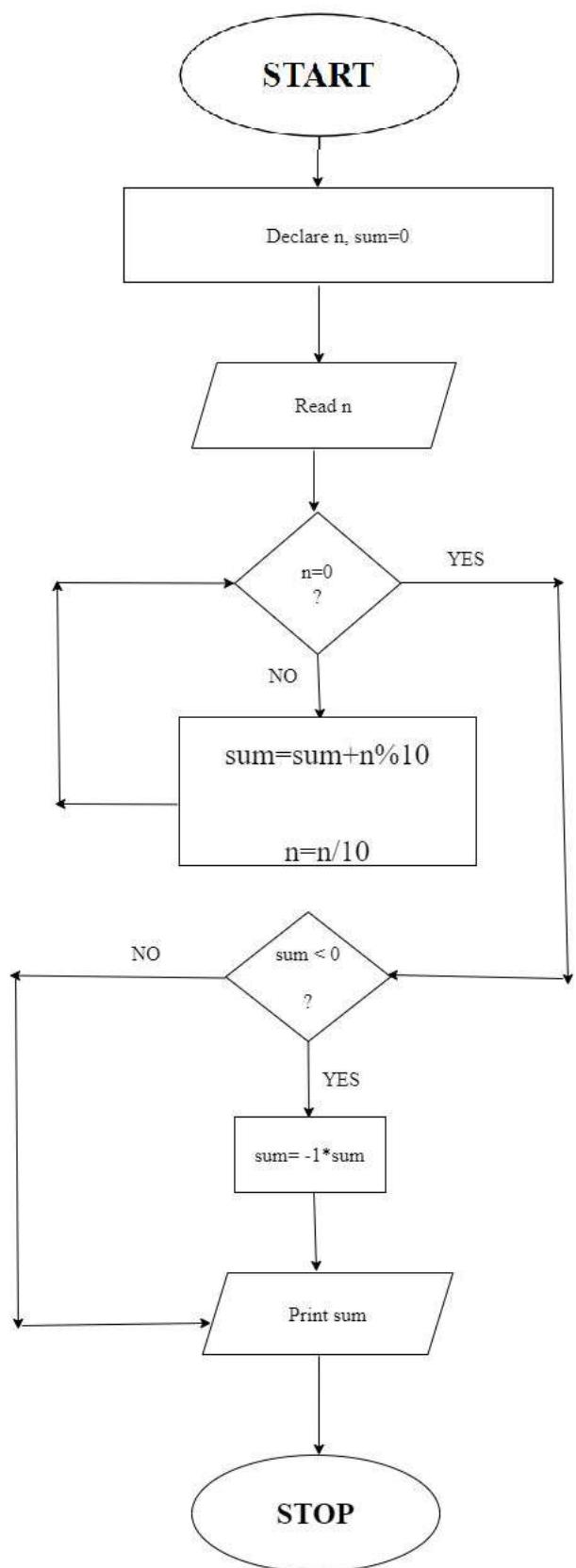
enter integer=-98706
sum of individual digits of the given integer is 30

...Program finished with exit code 0
Press ENTER to exit console. []

```

Inference- As the sum for negative integers is negative we need to take the absolute value of the result.

- a)
1. START
 2. Read n
 3. while $n! = 0$ do
 4. sum \leftarrow sum + $n \% 10$
 5. $n = n / 10$
 6. If sum < 0
 7. sum $\leftarrow (-1)^* \text{sum}$
 8. Print sum
 9. Else
 10. Print sum
 11. STOP



b)To check whether given number is prime or composite

```
#include <stdio.h>

int main() {
    int n, i, flag = 0;
    printf("Enter a positive integer= ");
    scanf("%d", &n);

    for (i = 2; i <= n / 2; ++i) {
        if (n % i == 0) {
            flag = 1;
            break;
        }
    }

    if (n == 1) {
        printf("1 is neither prime nor composite.");
    }
    else {
        if (flag == 0)
            printf("%d is a prime number.", n);
        else
            printf("%d is a composite number.", n);
    }
}
```

Samuela Abigail Mathew
71762108039

```
}
```

```
return 0;
```

```
}
```

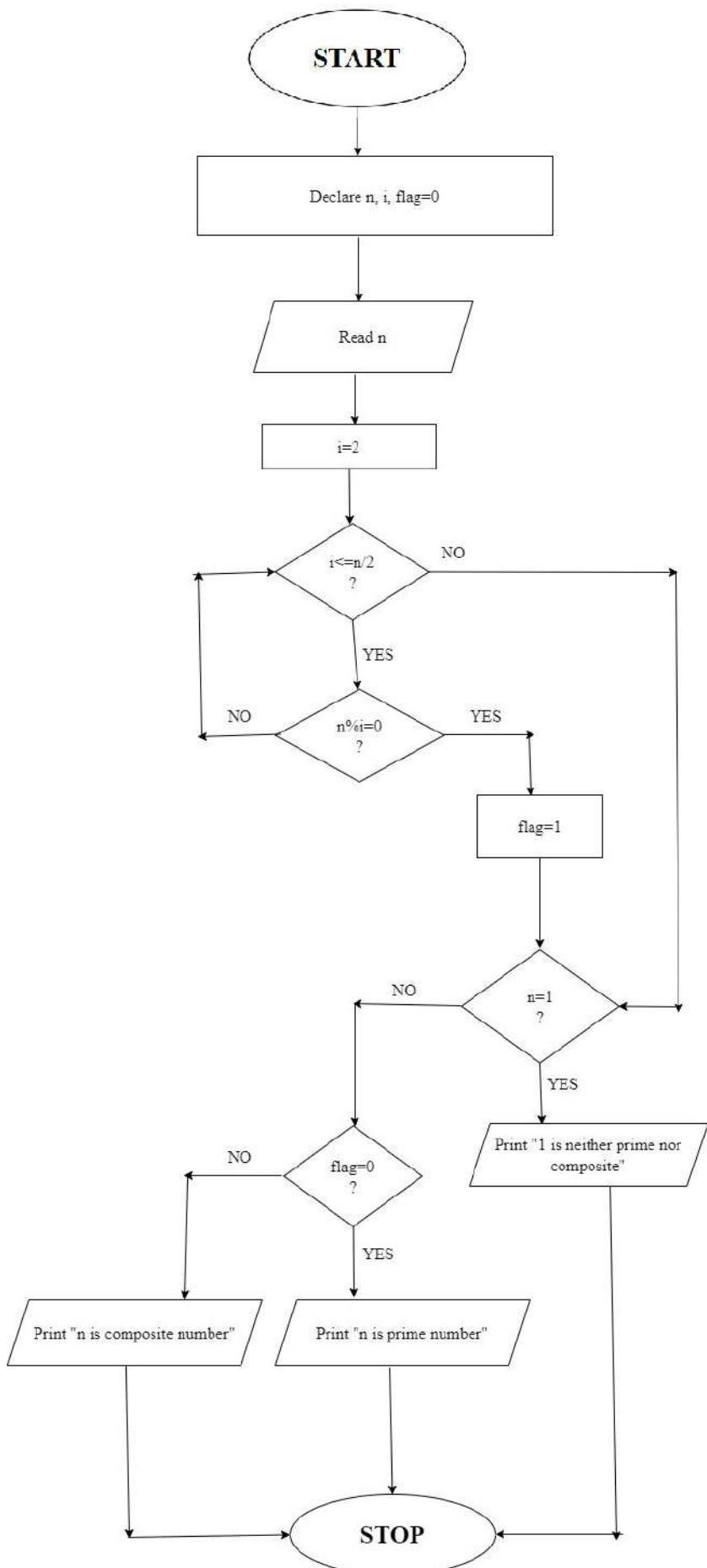
```
main.c
```

```
56 is a composite number.
```

```
...Program finished with exit code 0
Press ENTER to exit console.[]
```

Inference- The program does not check whether the number entered is zero or less than zero. Hence it states zero and all negative numbers as prime number.

b) 1. START
2. Read n
3. for $i=2$ to $i < n/2$ do
4. If $n \div i = 0$
5. flag $\leftarrow 1$
6. break
7. If $n < 1$
8. Print "1 is neither prime no
composite"
9. Else
10. If flag ≤ 0
11. Print "n is prime number"
12. Else
13. Print "n is composite number"
14. end for
15. $i > n/2$
16. STOP



c) To calculate square of numbers whose least significant digit is 5.

```
#include<stdio.h>

int main()
{
    int n, s, ls;
    printf("Enter the number whose least significant digit is 5:
\n");
    scanf("%d",&n);
    ls=n%10;
    s = n*n;

    {
        if(ls== 5)
            printf("Square of the number is = %d",s);

        else
            printf ("Invalid number");
    }
    return 0;
}
```

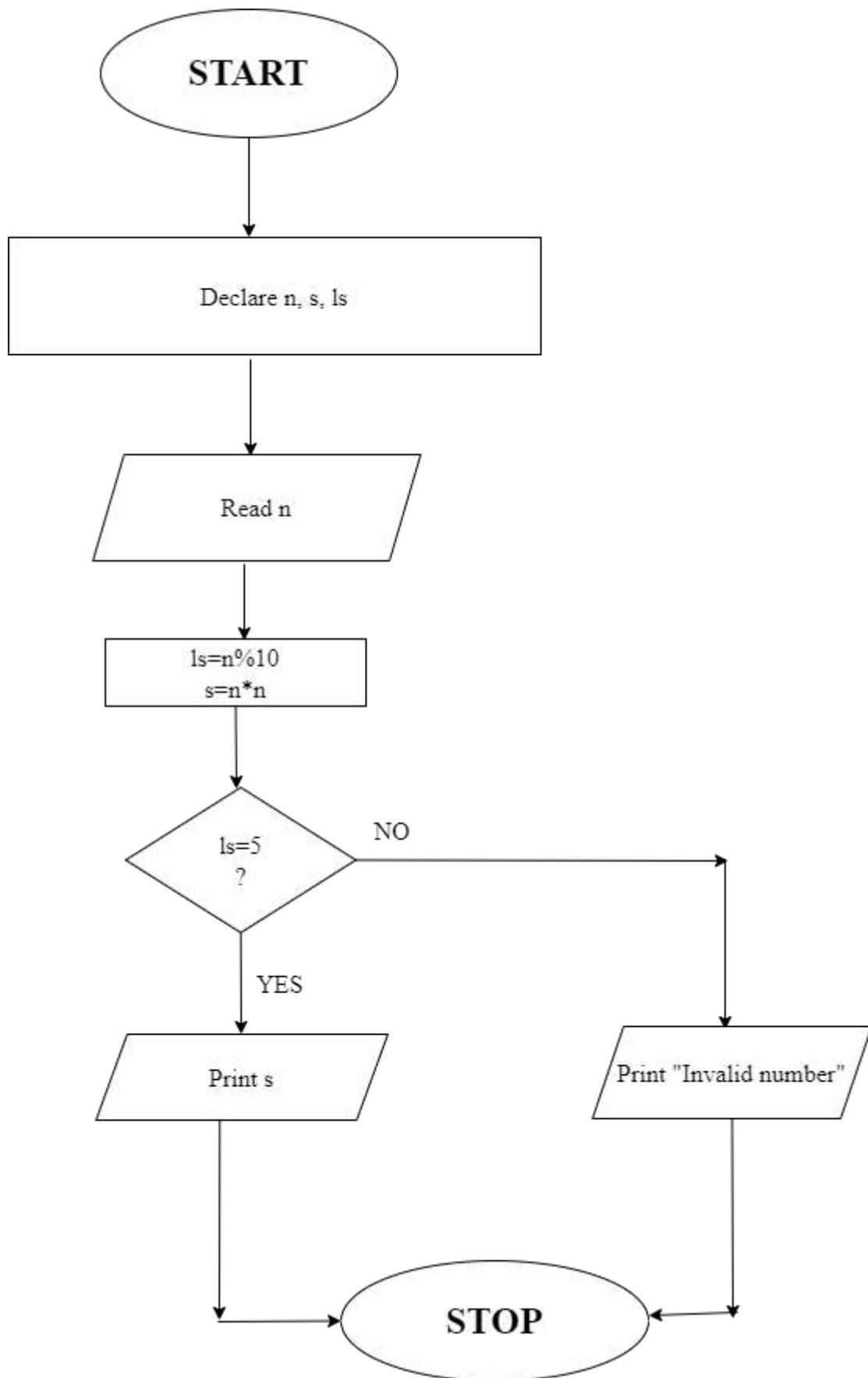
```
#include<stdio.h>
int main()
{
    int n;
    int ls;
    ls = n % 10;
    if (ls <= 5)
        printf("%d", n * n);
    else
        printf("Invalid number");
}
```

Enter the number whose least significant digit is 5:
6785
Square of the number is = 46036225

...Program finished with exit code 0
Press ENTER to exit console.

Inference- The program does not work for other numbers like floating point numbers with least significant digit as 5. Works well for integers though.

9) 1. START
2. Read n
3. ls < n % 10
4. s <= n * n
5. If ls <= 5
6. Print s
7. Else
8. Print "Invalid number"
9. STOP



d)Padovan sequence

```
#include <stdio.h>

int main() {

    int i, n, t1 = 1, t2 = 1, t3=1;
    int nextTerm = t1 + t2;

    printf("Enter the number of terms: ");
    scanf("%d", &n);

    printf("Padovan Series: %d, %d, %d ", t1, t2, t3);
    // print 4th to nth terms
    for (i = 3; i <= n; ++i) {
        printf(",%d ", nextTerm);
        t1 = t2;
        t2 = t3;
        t3 = nextTerm;
        nextTerm = t1 + t2;
    }
    return 0;
}
```

The screenshot shows a C IDE interface with the following details:

- Toolbar:** Includes icons for File, Save, Run, Debug, Stop, Share, and Save.
- File List:** Shows "main.c" is the active file.
- Code Editor:** Displays the following C code:

```
1 #include <stdio.h>
2 int main() {
3
4     int i, n, t1 = 1, t2 = 1, t3=1;
```
- Output Console:** Shows the program's execution:

```
Enter the number of terms: 5
Padovan Series: 1, 1, 1 ,2 ,2 ,3
...Program finished with exit code 0
Press ENTER to exit console.
```

Inference-

Enter the number of terms: 1

Padovan Series: 1, 1, 1

Enter the number of terms: 2

Padovan Series: 1, 1, 1

Enter the number of terms: 3

Padovan Series: 1, 1, 1 ,2

Enter the number of terms: 4

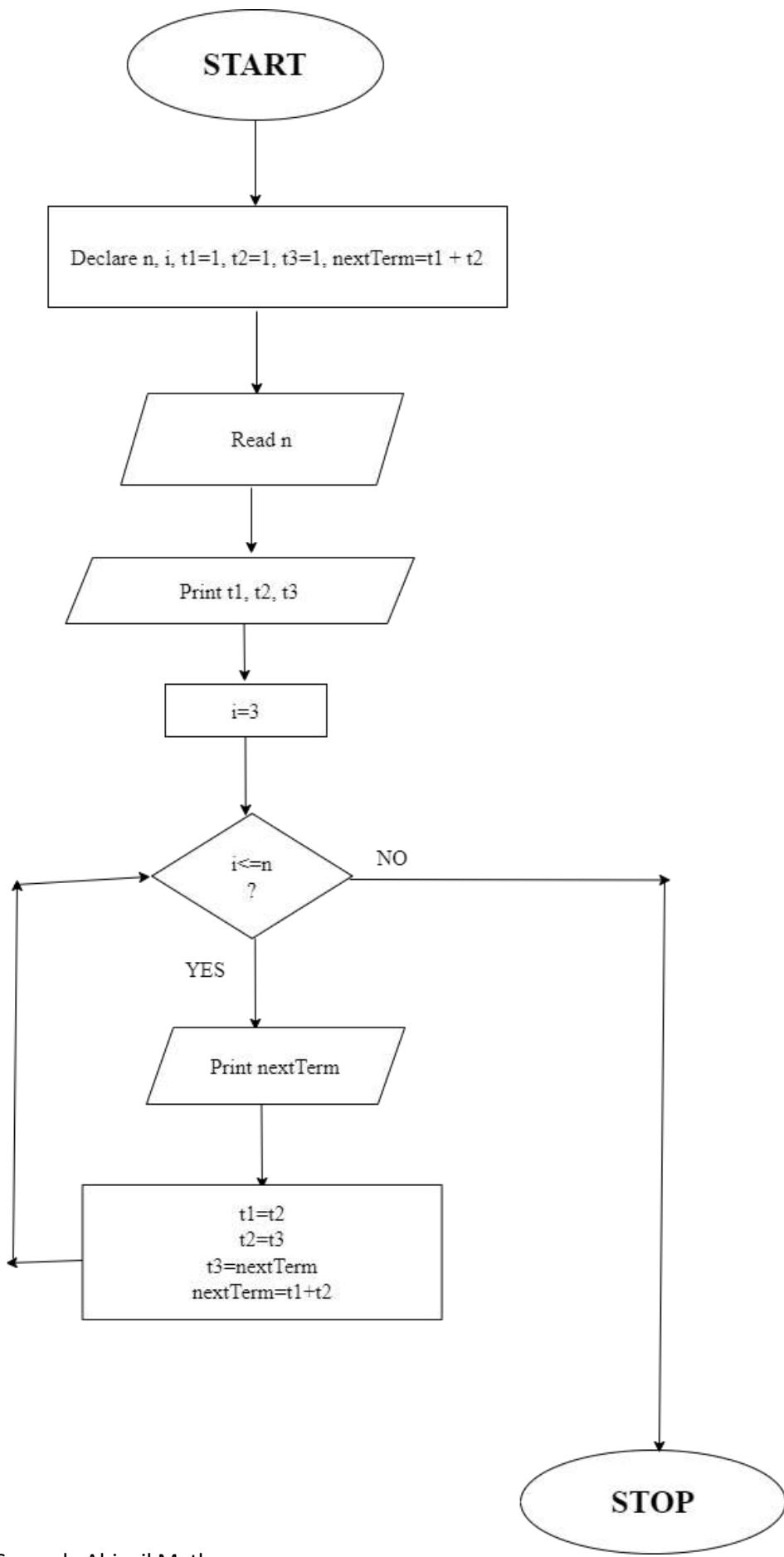
Padovan Series: 1, 1, 1 ,2 ,2

Enter the number of terms: 5

Padovan Series: 1, 1, 1 ,2 ,2 ,3

The number of terms in the series is not generated correctly.

d) 1. START
2. Read n
3. Print t_1, t_2, t_3
4. for $i \leq 3$ to n do
5. Print nextTerm
6. $t_1 \leftarrow t_2$
7. $t_2 \leftarrow t_3$
8. $t_3 \leftarrow$ nextTerm
9. nextTerm $\leftarrow t_1 + t_2$
10. end for
11. $i > n$
12. STOP



e)Find out the sum of series $1^2 + 2^2 + \dots + n^2$

```
#include<stdio.h>

int main()
{
    int n, sum;
    printf("Enter the number of terms in series: ");
    scanf("%d",&n);

    sum = (n * (n + 1) * (2*n + 1)) / 6;
    printf("Sum of the series is %d ", sum);

    return 0;
}
```

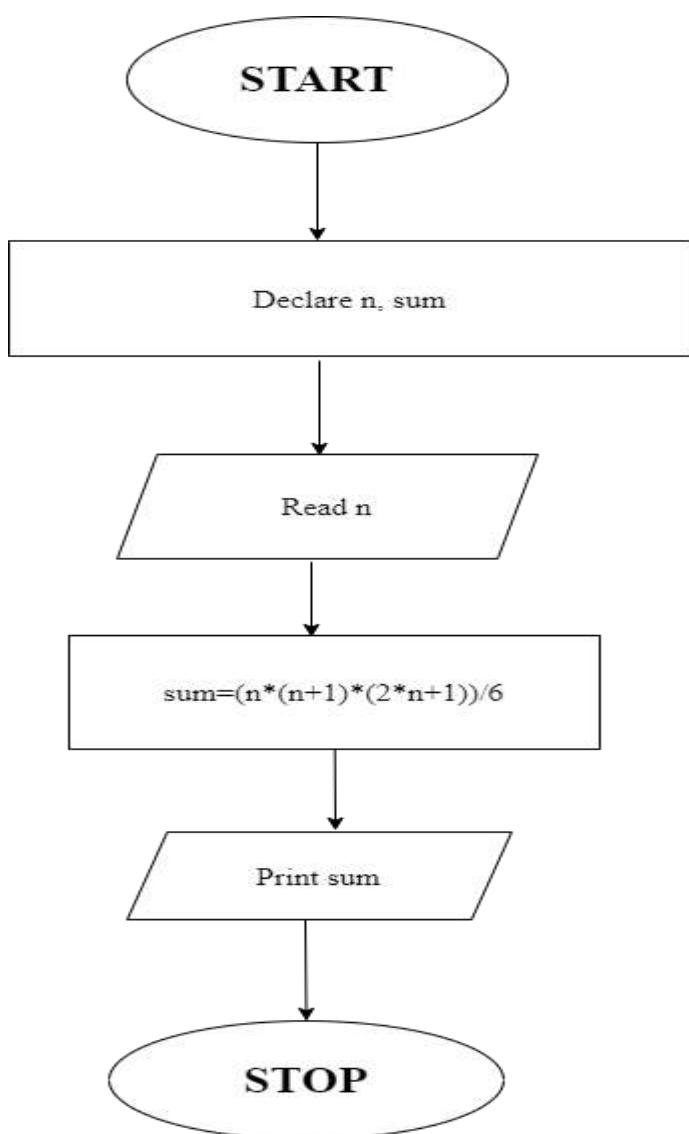
```
main.c
1 #include<stdio.h>
2 int main()
3 {
4     int n, sum;
5
6     printf("Enter the number of terms in series: ");
7     scanf("%d",&n);
8
9     sum = (n * (n + 1) * (2*n + 1)) / 6;
10    printf("Sum of the series is %d ", sum);
11
12    return 0;
13 }
```

```
Enter the number of terms in series: 5
Sum of the series is 55

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- The program does not handle negative numbers.

e) 1. START
2. Read n
3. sum $\leftarrow (n * (n + 1) * (2n + 1)) / 6$
4. Print sum
5. STOP



LAB 2- Take Home

Implement C programs for the following problem

statements:

a) Sine Series: Write a program that accepts x, and a number n ($n = 5$) and computes $\sin(x)$ using the sine series up to first n terms.

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

int main()
{
    int i, j, n=5, fact, sign = - 1;
    float x, p, sum = 0;
    float radianx=0;
    printf("Enter the value of x : ");
    scanf("%f", &x);
    radianx = x*3.14159/180.0;
    for (i = 1; i <= n; i++)
    {
        p = 1;
        fact = 1;
        for (j = 1; j <= i; j++)
        {
            fact = fact * j;
        }
    }
}
```

Samuela Abigail Mathew
71762108039

```

p=pow(radianx,i);

sign = - 1 * sign;

sum += sign * p / fact;

}

printf("sin %f = %.2f", x, sum);

return 0;

}

```

```

main.c
1 #include <stdio.h>
2 #include <math.h>
3
4 radians = x * 3.14159 / 180.0
5
6 fact = 1
7 for i=1 to n do
8   fact = fact * i
9 end for
10 p = pow(radians, i)
11 sign = -1 * sign
12 sum = sum + sign * p / fact
13 end for
14 printf "%f, %f", x, sum
15 STOP

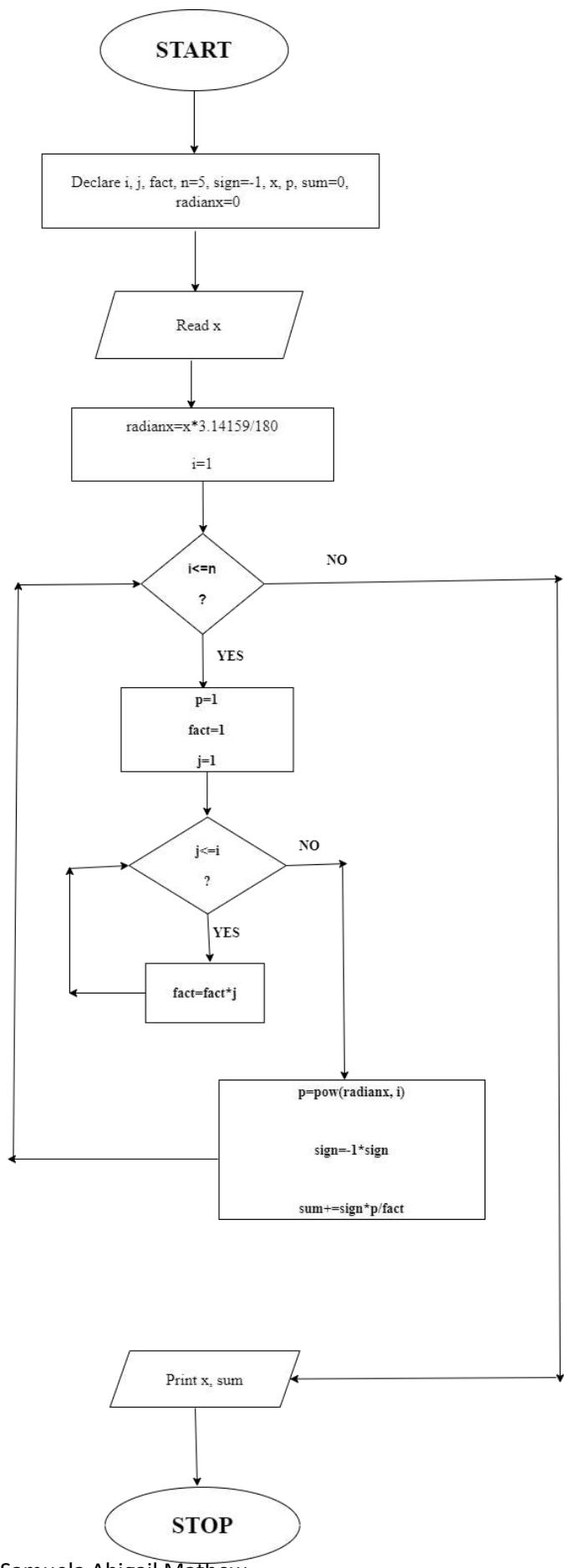
Enter the value of x : 4
sin 4.000000 = 0.07

...Program finished with exit code 0
Press ENTER to exit console.

```

Inference- It works well for negative integers and decimals also.

a) 1. START
2. read x
3. radianx = $x * 3.14159 / 180.0$
4. for i<=1 to n do
5. p<-1
6. fact<-1
7. for j<=1 to i do
8. fact = fact * j
9. end for
10. if j > i
11. p = pow(radianx, i)
12. sign = -1 * sign
13. sum = sum + sign * p / fact
14. end for
15. if i > n
16. printf x, sum
17. STOP



Samuela Abigail Mathew
71762108039

b) Second Largest: Input 5 integers, and display the second largest number.

```
#include <stdio.h>

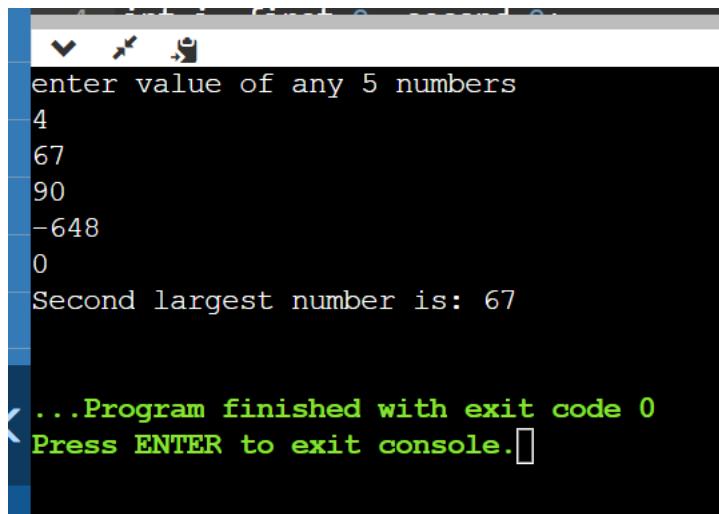
int main() {
    int array[5];
    int i, first=0, second=0;

    printf ("enter value of any 5 numbers \n");

    for(i = 0; i < 5; i++)
    {
        scanf("%d", &array[i]);
    }

    //first = array[0];
    for(i = 0; i < 5; i++)
    {
        if( first < array[i] )
        {
            second = first;
            first = array[i];
        }
        else if( second < array[i] )
```

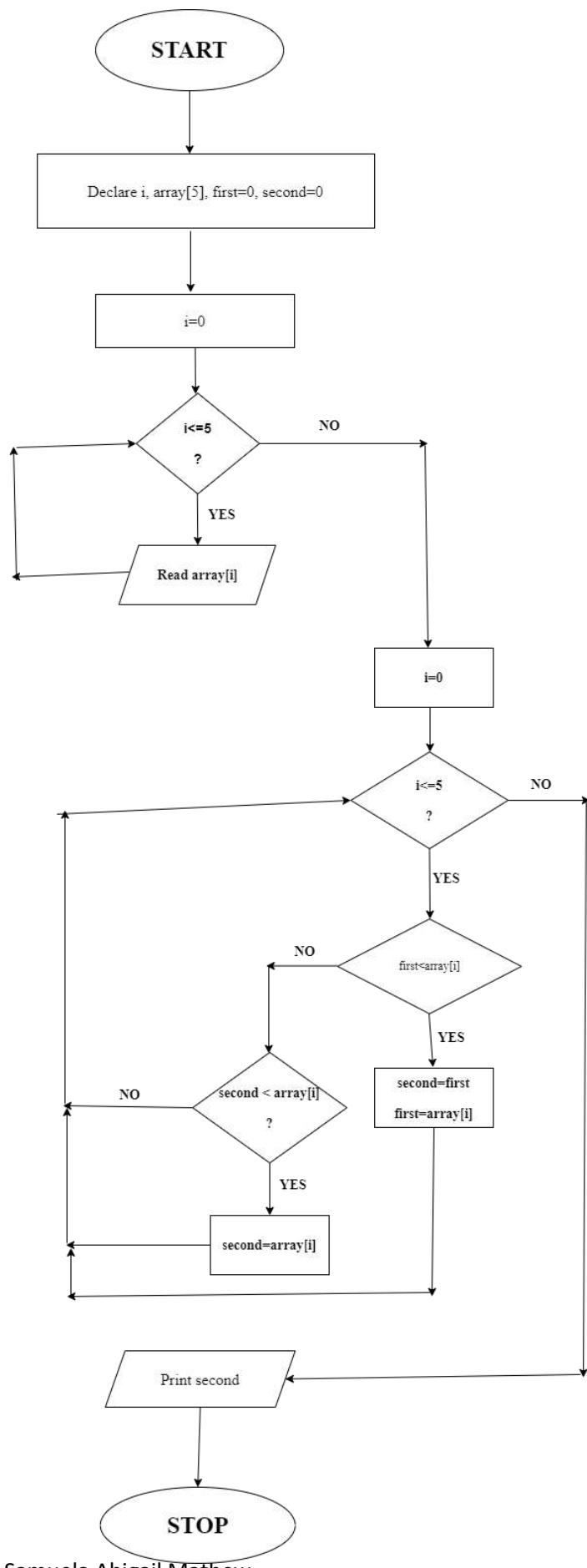
```
{  
    second = array[i];  
}  
}  
  
printf("Second largest number is: %d \n", second);  
  
return 0;  
}
```



```
enter value of any 5 numbers  
4  
67  
90  
-648  
0  
Second largest number is: 67  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Inference- It doesn't work for decimals. It doesn't work if all elements of the array are negative also.

b) 1. START
2. for $i \leftarrow 0$ to 5 do
3. Read array[i]
4. end for $i > 5$
5. for $i \leftarrow 0$ to 5 do
6. If first < array[i]
7. second = first
8. first = array[i]
9. Else if second < array[i]
10. second = array[i]
11. end for $i > 5$
12. Print second
13. STOP



c) Reversing a number: Input a four-digit number 'x' and print its digits in the reverse order.

```
#include <stdio.h>

int main() {
    int n, rev = 0, remainder;
    printf("Enter an integer: ");
    scanf("%d", &n);

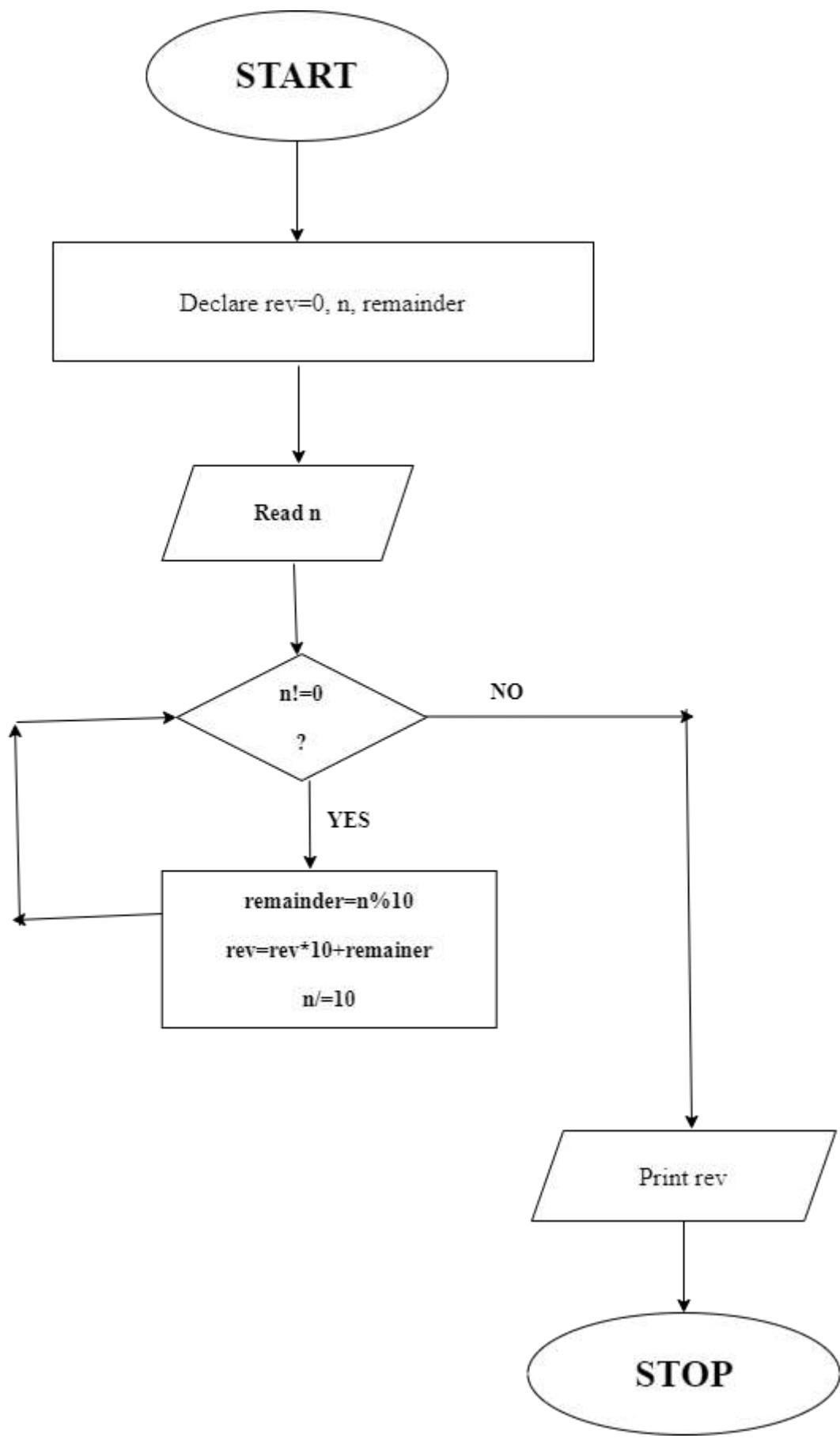
    while (n != 0)
    {
        remainder = n % 10;
        rev = rev * 10 + remainder;
        n /= 10;

    }
    printf("Reversed number = %d", rev);
    return 0;
}
```

```
main.c
1 #include <stdio.h>
2 int main() {
3     int n, rev = 0, remainder;
4     ...
5 }
6
7 Enter an integer: 67
8 Reversed number = 76
9
10 ...Program finished with exit code 0
11 Press ENTER to exit console.
```

Inference- The programs codes for reversing the digits of the given number. Works well for negative integers and numbers having first digit as zero also like -9805 and 0567 respectively.

1. START
2. Read n
3. while $n \neq 0$ do
4. remainder $\leftarrow n \% 10$
5. rev $\leftarrow rev * 10 + remainder$
6. $n \leftarrow n / 10$
7. Print rev
8. STOP



d) Division: Input a 4-digit number 'x' and check if it is divisible by one or more of the following seed numbers 2, 3, 4 and 12 or not. Use an effective strategy that inspects digits.

```
#include <stdio.h>

int main()
{
    int num, i, flag =0;
    printf("Enter any 4 digit number: ");
    scanf("%d", &num);
    if ((num<1000) || (num>9999) )
    {
        printf("< %d > is not a 4 digit number", num);
        return 0;
    }
    int array[4] = {2,3,4,12};
    for (i=0;i<4; i++)
    {
        if (num % array[i] == 0)
        {
            printf("< %d > is seed number\n", array[i]);
            flag = 1;
        }
    }
}
```

Samuela Abigail Mathew
71762108039

```

if (flag==0)

printf("Number is not divisible by any seed number");

return 0;

}

```

```

main.c
5 printf( "Enter any 4 digit number. " );
6 scanf("%d", &num);
7 if ((num<1000) || (num>9999) )
8 {

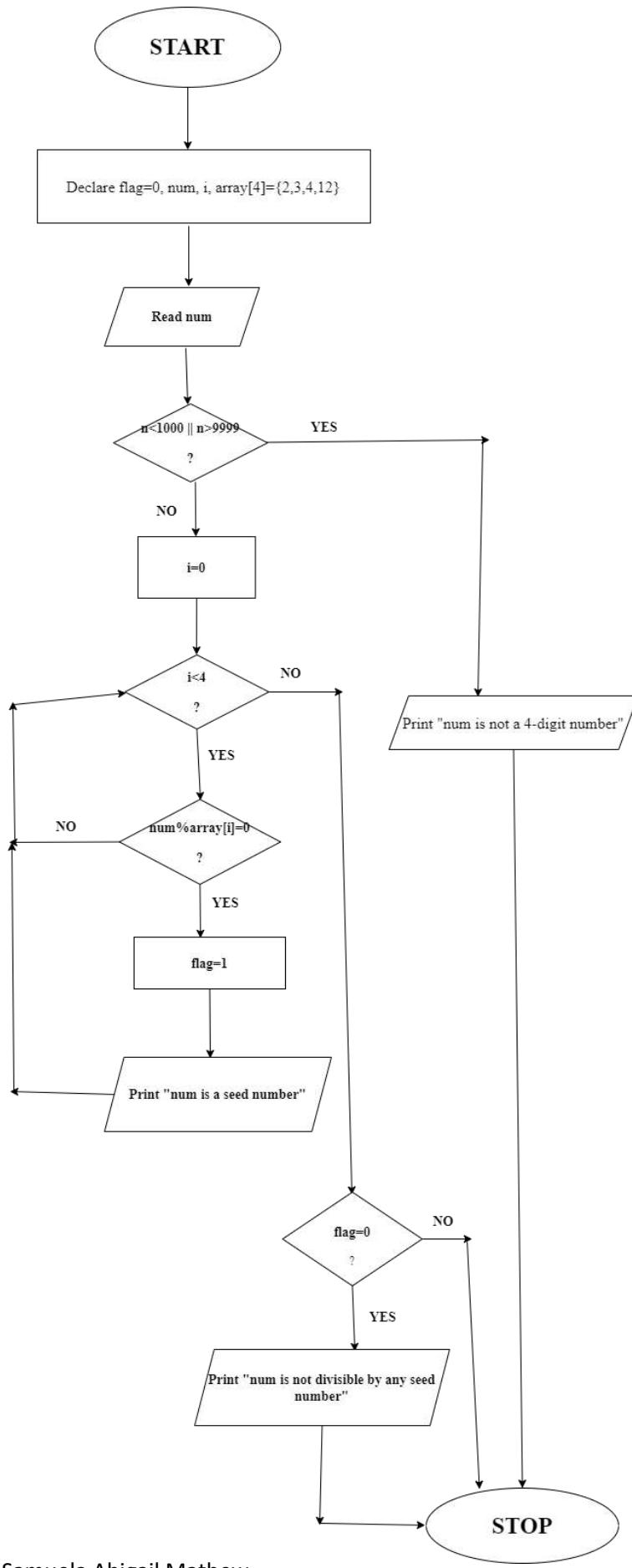
Enter any 4 digit number: 5638
< 2 > is seed number

...Program finished with exit code 0
Press ENTER to exit console. []

```

Inference- it doesn't work for numbers not having 4 digits. It won't work for decimals also.

1. START
 2. Read num
 3. If num < 1000 OR num > 9999
 4. Print "<num> is not a 4 digit number"
 5. Exit
 6. for i=0 to 4 do
 7. If num % array[i] == 0
 8. Print "<array[i]> is seed number"
 9. flag = 1
 10. If flag == 0
 11. Print "<num> not divisible by any seed number"
 12. STOP



LAB 3

Aim- Analyze code snippets and correct them using control statements.

Analyze the following code snippets and give your inferences

a) What will be the value of j for below-mentioned values
of i?

```
switch (i) {  
    case 2: i = i * i; case 4: i = i * i; default: i = i * i;  
    break;  
    case 16: i = i * i;}j = i;  
  
#include <stdio.h>  
  
int main()  
{  
    int i = 1, j;  
    printf("i is %d \n", i);  
    switch (i) {  
        case 2: i = i * i;  
        case 4: i = i * i;  
        default: i = i * i;
```

```

break;

case 16: i = i * i;

}

j = i;

printf("j is %d \n", j);

return 0;

}

```

The screenshot shows a terminal window with the following content:

```

main.c
1 #include <stdio.h>
2 int main()
3 {
4     int i, j;
5     i = 1;
6     j = i * i;
7     printf("i is %d\n", i);
8     printf("j is %d\n", j);
9 }
10
11 ...Program finished with exit code 0
12 Press ENTER to exit console. []

```

Inference-

For $i = 2, j = 256$

For $i = 4, j = 256$

For $i = 16, j = 256$

For $i = 1, j = 1$

b) What would be the output of the following program

```
#define m 5+5 const int n = 5+5;  
  
void main() {  
  
    int a = 0, b = 0;  
  
    a = m * m; b = n * n;  
  
    printf("%d %d\n", a, b);  
  
}
```

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

```
#define m 5+5
```

```
const int n = 5+5;
```

```
void main() {
```

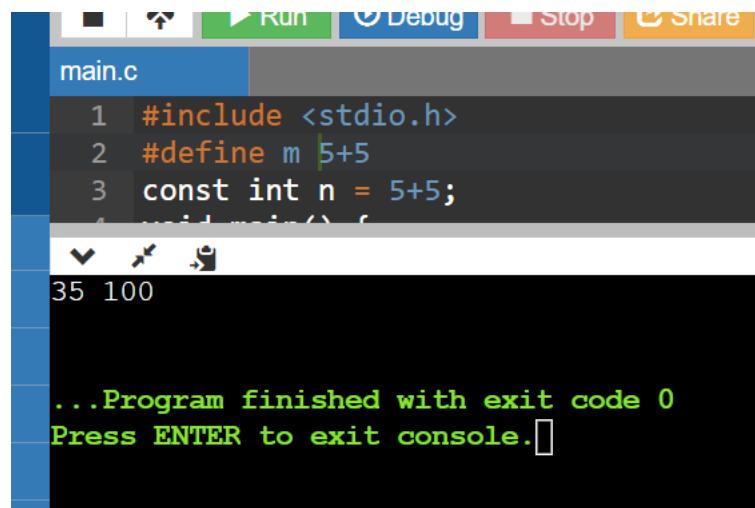
```
int a = 0, b = 0;
```

```
a = m * m;
```

`b = n * n;`

```
printf("%d %d\n", a, b);
```

}



Inference- Value of a is 35 and b is 100 because m will be considered as 5+5, whereas n=5+5=10. So a=5+5*5+5=5+25+5=35, whereas b=10*10=100

c) What would be the output of the following program

```
#include <stdio.h>

int main()
{
    int i = 1, j = 1, k = 1, count = 0;

    while (i < 2) {
        for(; j < 4; j += k) do {
            ++count; k += i;
        } while (k < 8); i += j;
    }

    printf("Loop Indices: %d %d %d\n", i, j, k);
    printf("Number of iterations = %d\n", count);

    return 0;
}

#include <stdio.h>

int main()
{
    int i = 1, j = 1, k = 1, count = 0;
    while (i < 2) {
```

```

for(; j < 4; j += k) do {
    ++count; k += i;
}

while (k < 8);
i += j;
}

printf("Loop Indices: %d %d %d\n", i, j, k);
printf("Number of iterations = %d\n", count);
return 0;
}

```

```

main.c
1 #include <stdio.h>
2 int main()
3 {
4     ...
5 }
6
7 Loop Indices: 10 9 8
8 Number of iterations = 7
9
10 ...Program finished with exit code 0
11 Press ENTER to exit console. []

```

Inference- Loop Indices are 10, 9, and 8 respectively whereas Number of iterations is 7. This is because inside while loop, for loop will execute the do-while loop only. After for loop ends, $i+=j$ statement is executed.

d) Implement C programs for the following problem statements:

Fibonacci Series: Fibonacci numbers are the numbers in the following

Integer sequence: 0,1,1,2,3,5,8,13,21 ... By definition, the first two Fibonacci numbers are 0 and 1, and each subsequent number is the sum of the previous two numbers. Write a program that accepts a set of 'k' seed values, and find the Fibonacci series for n with seed values $\text{fib}(n) = \text{fib}(n - 1) + \text{fib}(n - k)$

```
#include <stdio.h>

int main() {
    int n, k=0, N;
    printf("Enter the number of terms to generate in the fibonacci series:
    ");
    scanf("%d", &N);
    printf("Enter the value of k: ");
    scanf("%d", &k);

    int fib[k+2+N];
    fib[0]=0;
    fib[1]=1;

    printf("Enter set of %d seed values: \n", k);
    for (n=2; n<k+2; n++)
    {
        scanf("%d", &fib[n]);
    }
```

```

printf("The fibonacci series is 0");
for (n=1; n<k+2; n++)
{
printf(", %d",fib[n]);
}
k=k+2; //Since fibonacci starts with predefined values 0 and 1
for (n=k; n<=k+N-1; n++)
{
fib[n]= fib[n-1] + fib[n-k];
printf(", %d", fib[n]);
}
return 0;
}

```

```

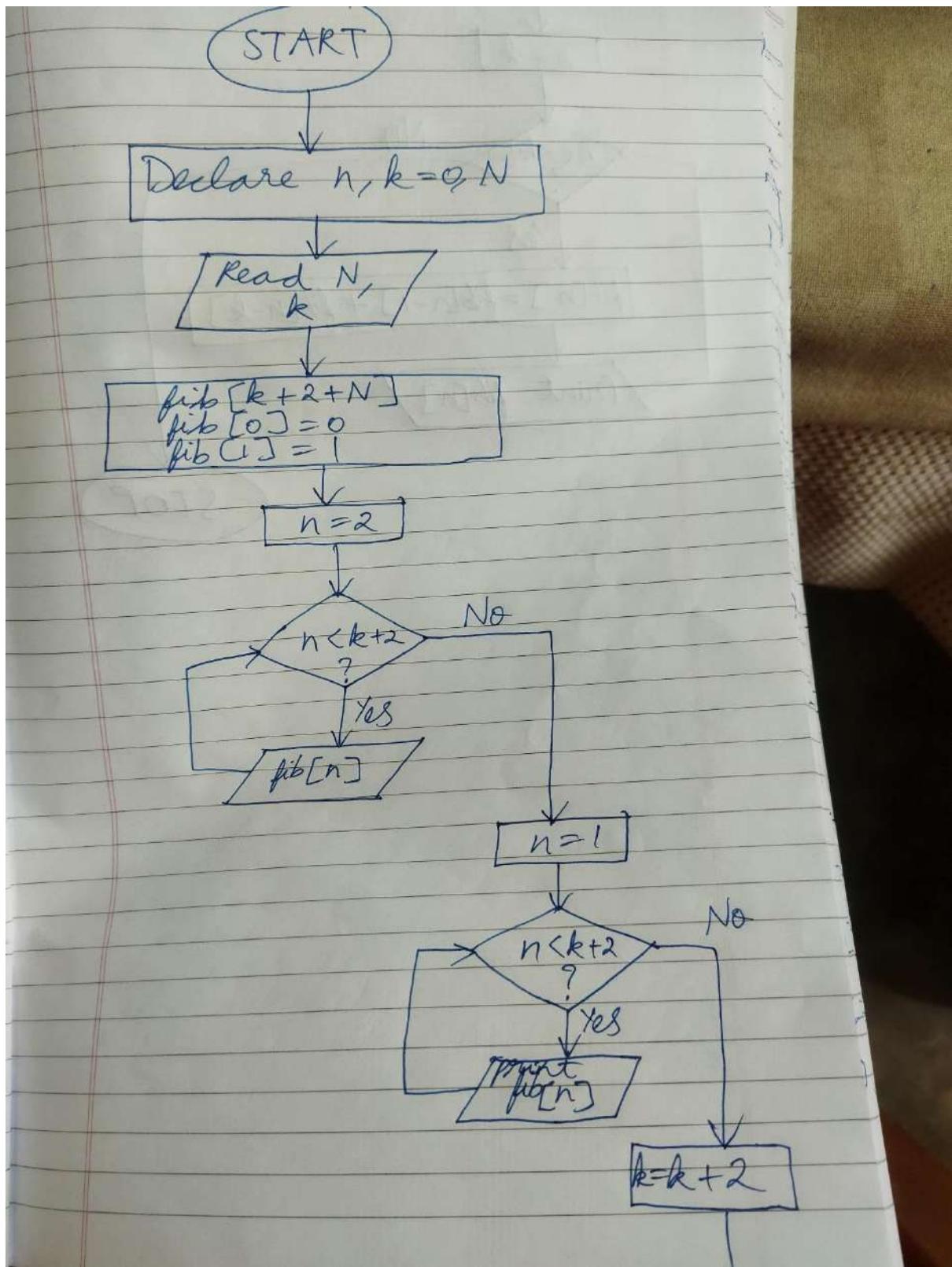
main.c
1 #include <stdio.h>
2 int main() {
3     int n, k=0, N;
4
5     printf("Enter the number of terms to generate in the fibonacci series: ");
6     scanf("%d", &n);
7     printf("Enter the value of k: ");
8     scanf("%d", &k);
9     printf("Enter set of 4 seed values:");
10    scanf("%d", &N);
11    scanf("%d", &fib[0]);
12    scanf("%d", &fib[1]);
13    scanf("%d", &fib[2]);
14    scanf("%d", &fib[3]);
15
16    for (n=k; n<=n+k-1; n++)
17    {
18        fib[n]= fib[n-1] + fib[n-k];
19        printf(", %d", fib[n]);
20    }
21
22    return 0;
23 }

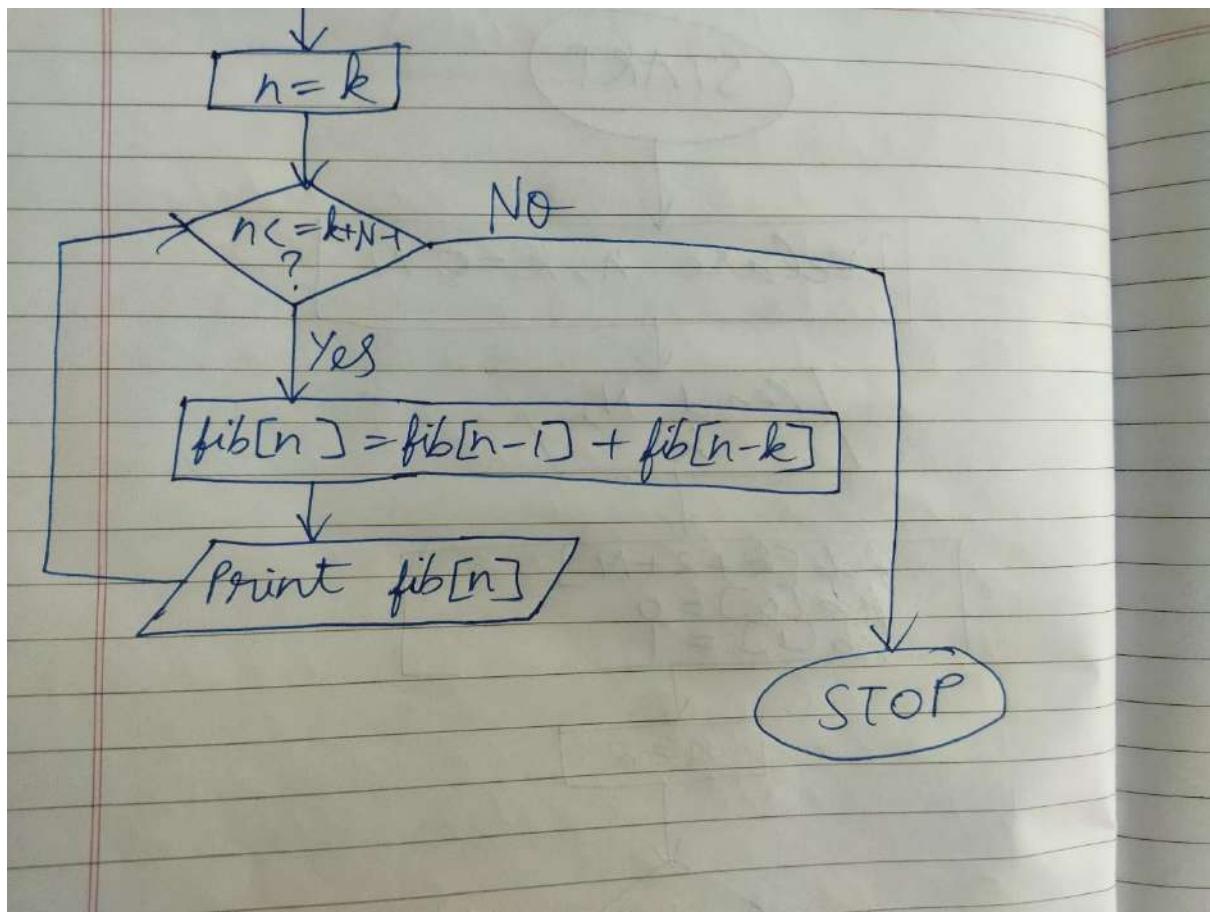
input
Enter the number of terms to generate in the fibonacci series: 11
Enter the value of k: 4
Enter set of 4 seed values:
2
3
8
5
The fibonacci series is 0, 1, 2, 3, 8, 5, 5, 6, 8, 11, 19, 24, 29, 35, 43, 54, 73
...Program finished with exit code 0
Press ENTER to exit console. []

```

Inference- First two values of Fibonacci series are predetermined as 0 and 1 respectively. It works only when $k < n$ since the term number can't be negative (from formula of Fibonacci series).

```
1. START
2. Read N, k
3. for n ← 2 to k+2 do
4.   Read fib[n]
5. end for n ← k+2
6. for n ← 1 to k+2 do
7.   print fib[n]
8. end for n ← k+2
9. k ← k+2
10. for n ← k to k+N-1 do
11.   fib[n] ← fib[n-1] + fib[n-k]
12.   Print fib[n]
13. end for 'n > k+N-1'
14. STOP
```





LAB 4

Aim- Learn to use control statements (part 2).

Implement C programs for the following problem statements:

1. Program to print words corresponding numbers below 9.

(Switch)

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

```
int main()
```

```
{
```

```
int n, num = 0;
```

```
printf("Enter any number: ");
```

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

```
while(n != 0)
```

```
{
```

```
num = (num * 10) + (n % 10);
```

```
n /= 10;
```

```
}
```

```
while(num != 0)
```

```
{
```

Samuela Abigail Mathew
71762108039

```
switch(num % 10)
```

```
{
```

```
case 0:
```

```
printf("Zero ");
```

```
break;
```

```
case 1:
```

```
printf("One ");
```

```
break;
```

```
case 2:
```

```
printf("Two ");
```

```
break;
```

```
case 3:
```

```
printf("Three ");
```

```
break;
```

```
case 4:
```

```
printf("Four ");
```

```
break;
```

```
case 5:
```

```
printf("Five ");
```

```
break;
```

```
case 6:
```

```
printf("Six ");
```

```
break;  
case 7:  
printf("Seven ");  
break;  
case 8:  
printf("Eight ");  
break;  
case 9:  
printf("Nine ");  
break;  
}  
  
num = num / 10;  
}
```

```
return 0;  
}
```

The screenshot shows a C IDE interface with the following details:

- Toolbar:** Run, Debug, Stop, Share, Save.
- Code Editor:** File named "main.c".

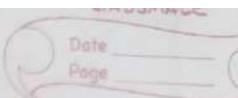
```
35 break;
36 case 6:
37 printf("Six ");
```
- Output Console:**

```
Enter any number: 1098
One Zero Nine Eight

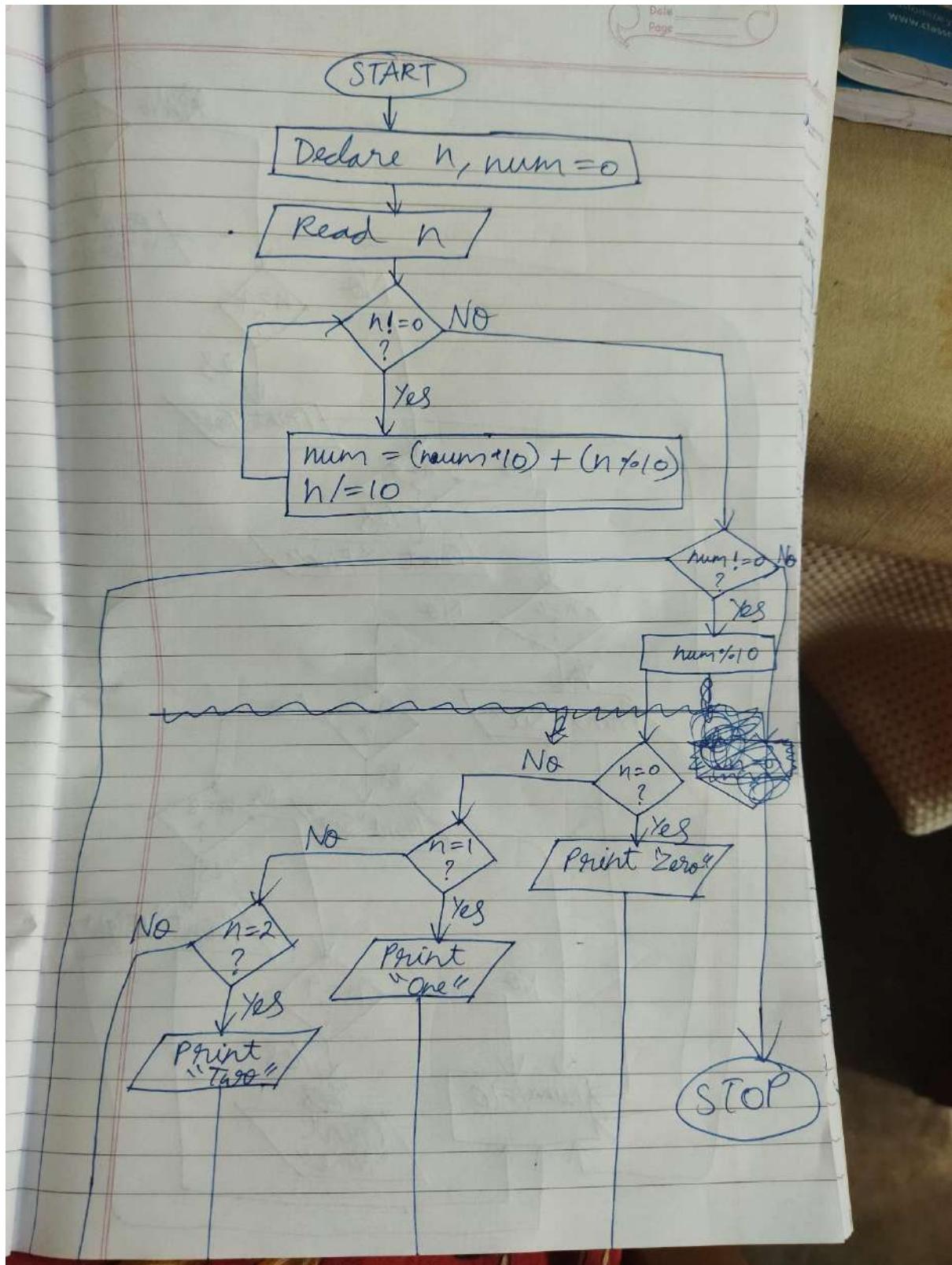
...Program finished with exit code 0
Press ENTER to exit console.[]
```

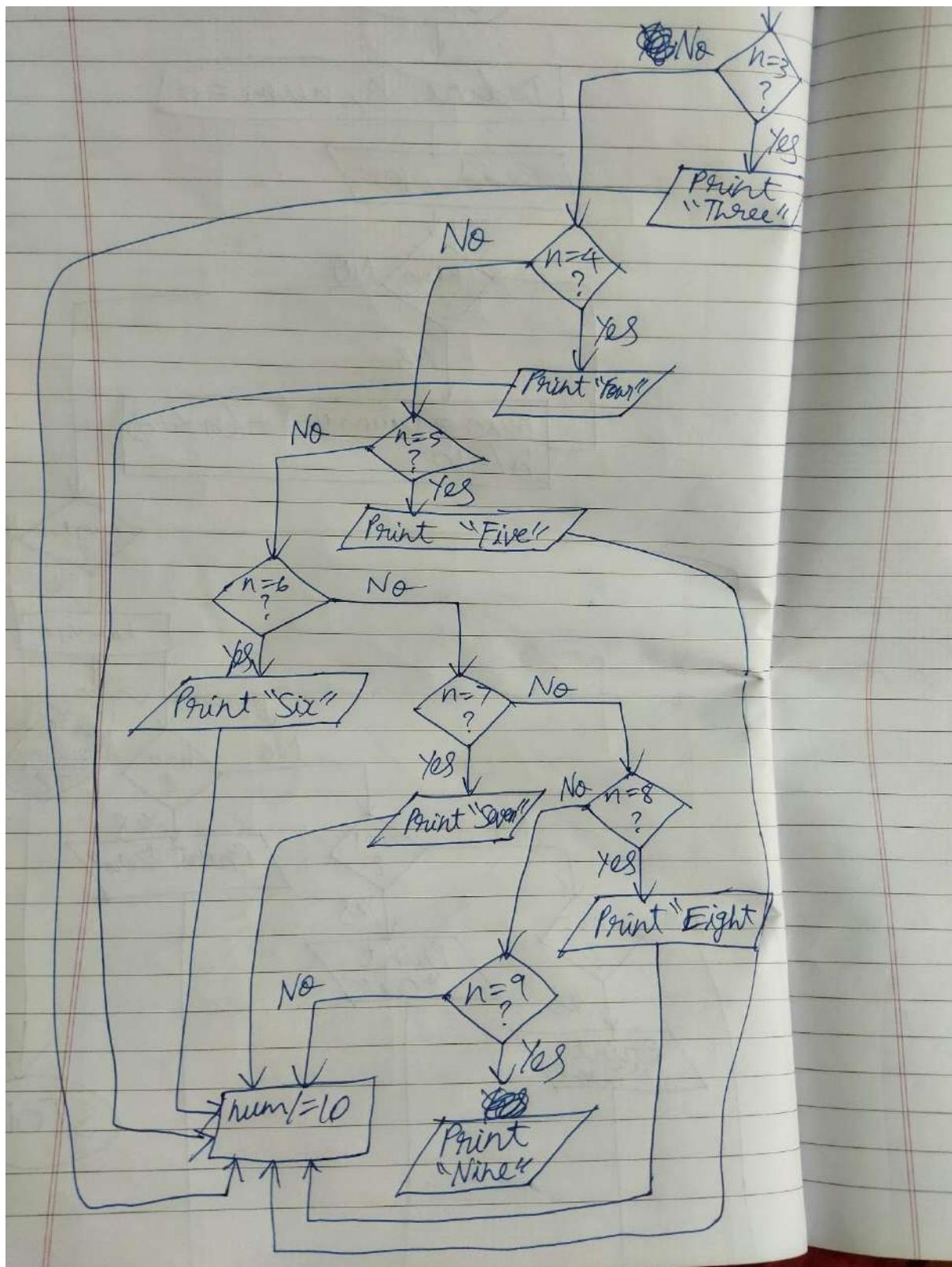
Inference- since we are asked to print numbers below 9, it can print number names in English for each digit.

1) 1. START
2. Read n
3. while $n \neq 0$
4. num = (num % 10) + (n // 10)
5. $n \leftarrow n / 10$
6. while num != 0
7. switch (num % 10)
8. case 0:
9. Print "zero"
10. case 1:
11. Print "One"
12. case 2:
13. Print "Two"
14. case 3:
15. Print "Three"
16. case 4:
17. Print "Four"



18. case 5:
19. Print "Five"
20. case 6:
21. Print "Six"
22. case 7:
23. Print "Seven"
24. case 8:
25. Print "Eight"
26. case 9:
27. Print "Nine"
28. num = num // 10
29. STOP





2. Program to calculate Arithmetic Operations depending on operator.

```
#include <stdio.h>

int main() {
    double a, b;
    char op;

    printf("Enter character of arithmetic operation (+, -, *, /): ");
    scanf("%c", &op);

    printf("Enter value of 2 numbers: ");
    scanf("%lf %lf", &a, &b);

    switch (op) {
        case '+':
            printf("Addition of %lf and %lf is %lf", a, b, a+b);
            break;

        case '-':
            printf("Subtraction of %lf and %lf is %lf", a, b, a-b);
            break;
    }
}
```

```
case '*':
printf("Multiplication of %lf and %lf is %lf", a, b,a*b);
break;

case '/':
printf("Division of %lf and %lf is %lf", a, b, a/b);
break;

default:
printf("Error! operator is not correct");
}

return 0;
}
```

The screenshot shows a code editor with a toolbar at the top containing icons for Run, Debug, Stop, Share, Save, and Beautify. The file 'main.c' is open, displaying the following code:

```
15 break,
16
17 case '-':
18 printf("Subtraction of %lf and %lf is %lf", a, b, a-b);
```

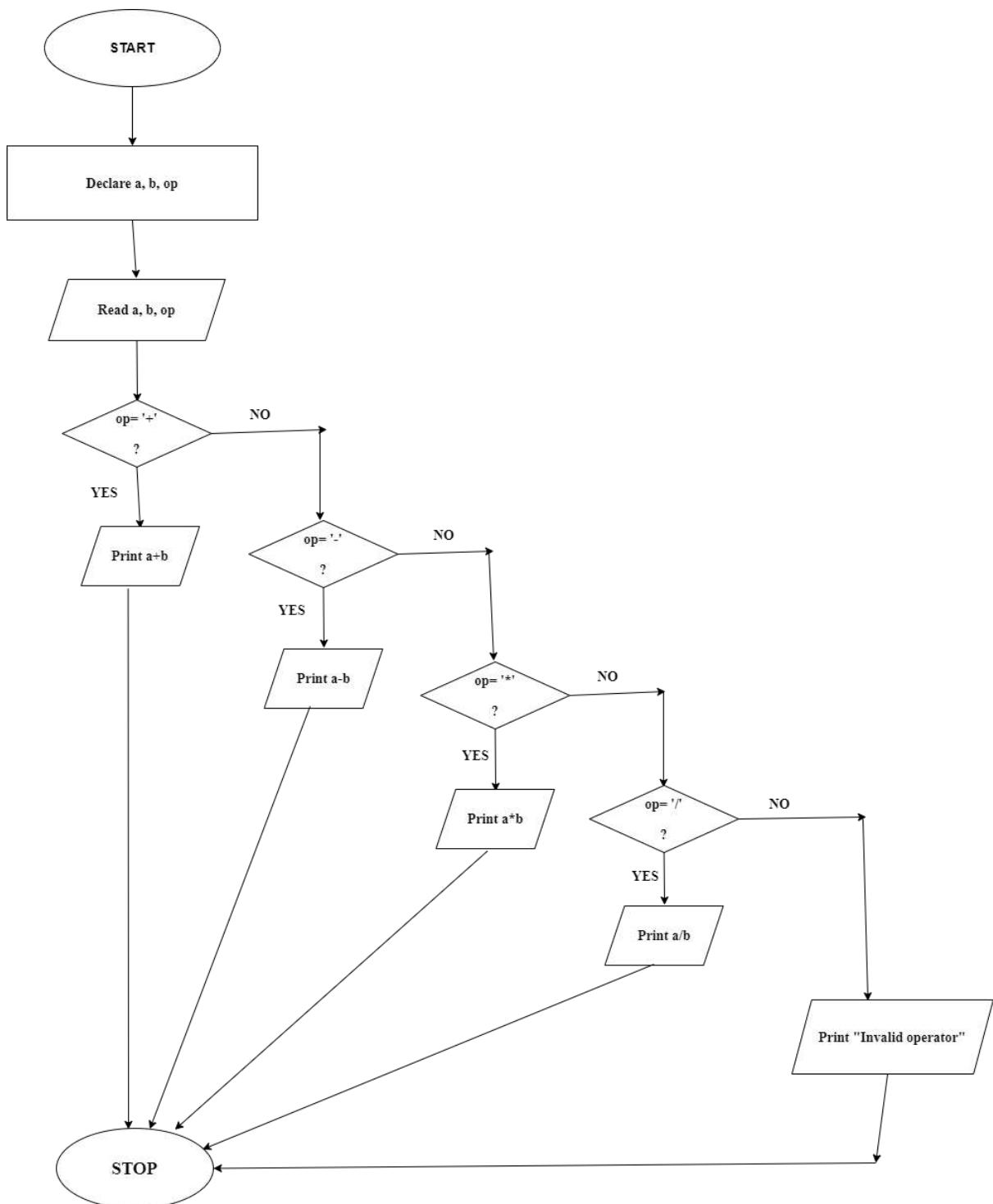
Below the code, there is a terminal window with the following output:

```
Enter character of arithmetic operation (+, -, *, /): /
Enter value of 2 numbers: 78 4
Division of 78.000000 and 4.000000 is 19.500000

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- switch case statement is used to code for a program that calculates in accordance to the arithmetic operator, but only a single operator can be acted on only two operands at a time.

b) 1. START
2. Read op, a, b
3. Switch (op)
4. case '+':
5. Print a+b
6. case '-':
7. Print a-b
8. case '*':
9. Print a*b
10. case '/':
11. Print a/b
12. default:
13. Print "operator is not correct"
14. STOP



3. Display numbers in the following format.

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

```
#include <stdio.h>

int main() {
    int i, j, rows;
    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    for (i = 1; i <= rows; ++i) {
        for (j = 1; j <= i; ++j) {
            printf("%d ", j);
        }
        printf("\n");
    }
    return 0;
}
```

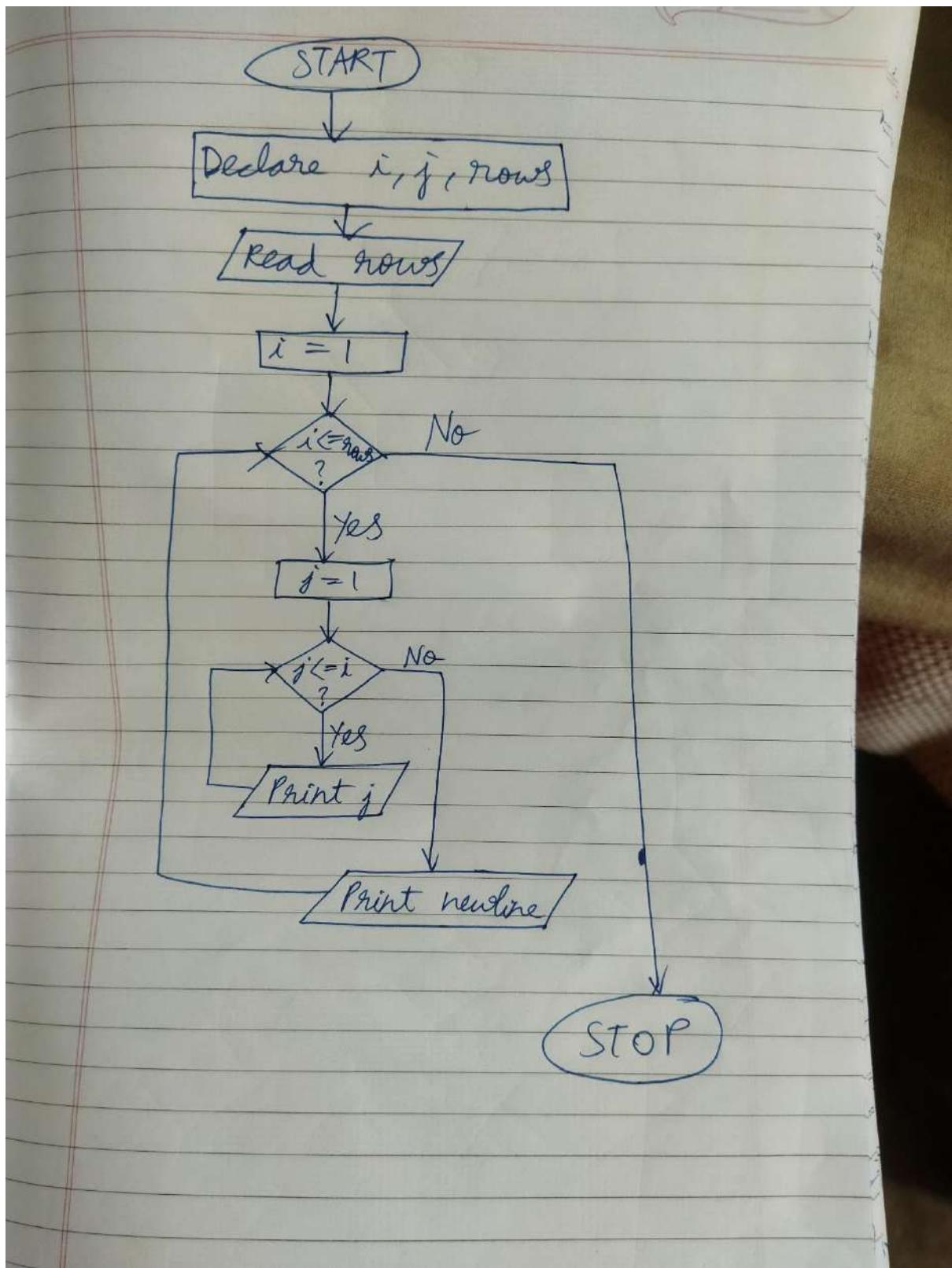
```
main.c
1 #include <stdio.h>
2 int main() {
3     int i, j, rows;
4     rows = 5;
5     for (i = 1; i <= rows; i++) {
6         for (j = 1; j <= i; j++) {
7             printf("%d ", j);
8         }
9         printf("\n");
10    }
11 }
```

```
Enter the number of rows: 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- the number pyramid is printed based on increment operand.

c) 1. START
2. Read rows
3. for $i \leq 1$ to rows do
4. for $j \leq 1$ to i do
5. Print j
6. end for $j > i$
7. Print " $\backslash n$ "
8. end for $i > rows$
9. STOP



4. Display numbers in the following format.

7 5 1 4 2

5 1 8 4

2 7 5

3 2

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

```
int main() {
```

```
    int i, j, k, l;
```

```
    int row1[5] = {7, 5, 1, 4, 2};
```

```
    int row2[4] = {5, 1, 8, 4};
```

```
    int row3[3] = {2, 7, 5};
```

```
    int row4[2] = {3, 2};
```

```
    for(i=0; i<5; i++)
```

```
    {
```

```
        printf("%d ", row1[i]);
```

```
    }
```

```
    printf("\n");
```

```
    for(j=0; j<4; j++)
```

```
    {
```

Samuela Abigail Mathew
71762108039

```
printf("%d ", row2[j]);  
}
```

```
printf("\n");
```

```
for(k=0; k <3; k++)
```

```
{
```

```
printf("%d ", row3[k]);
```

```
}
```

```
printf("\n");
```

```
for(l=0; l <2; l++)
```

```
{
```

```
printf("%d ", row4[l]);
```

```
}
```

```
return 0;
```

```
}
```

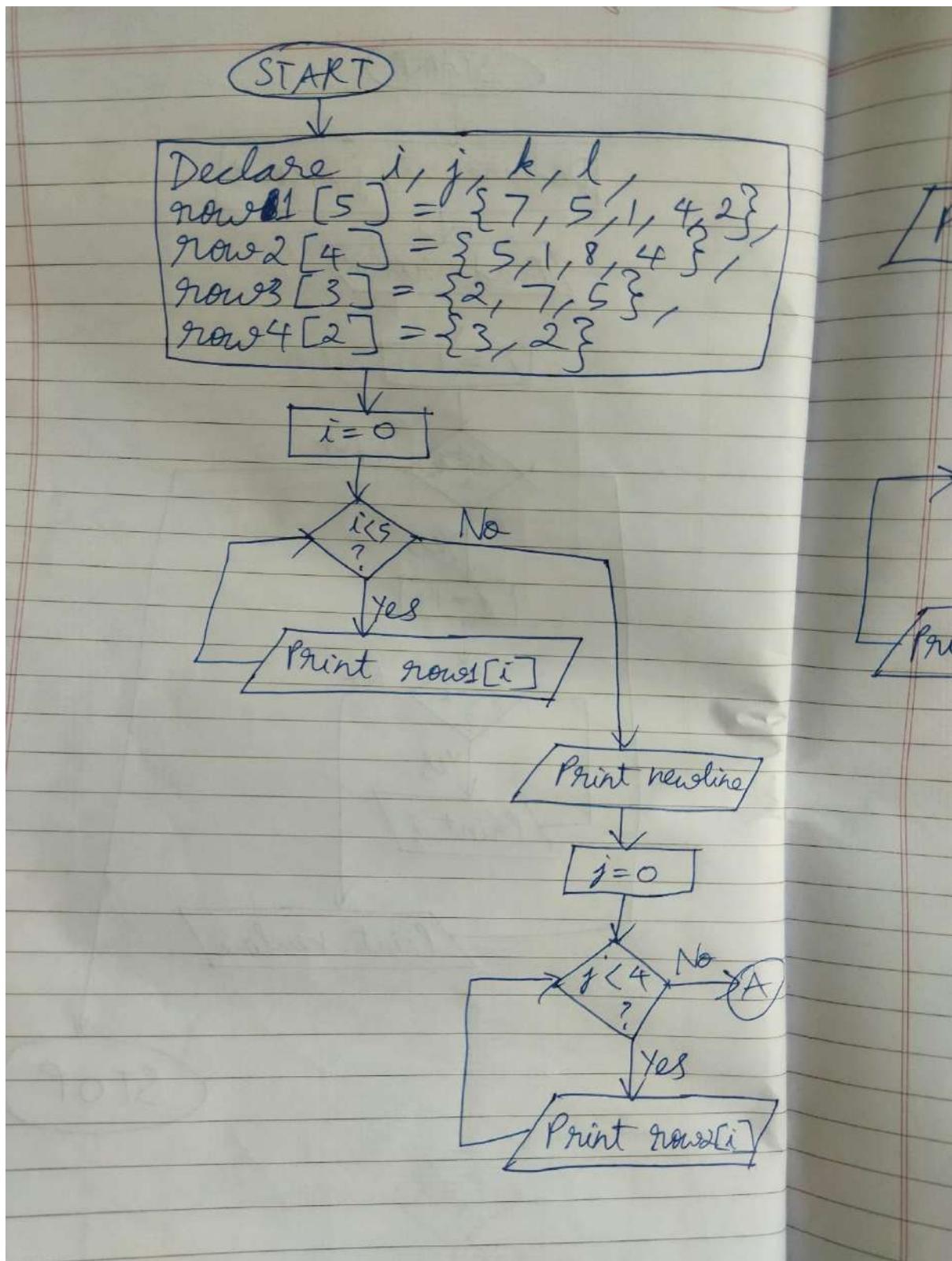
```
main.c
18     printf("%d ", row2[j]);
19 }
20

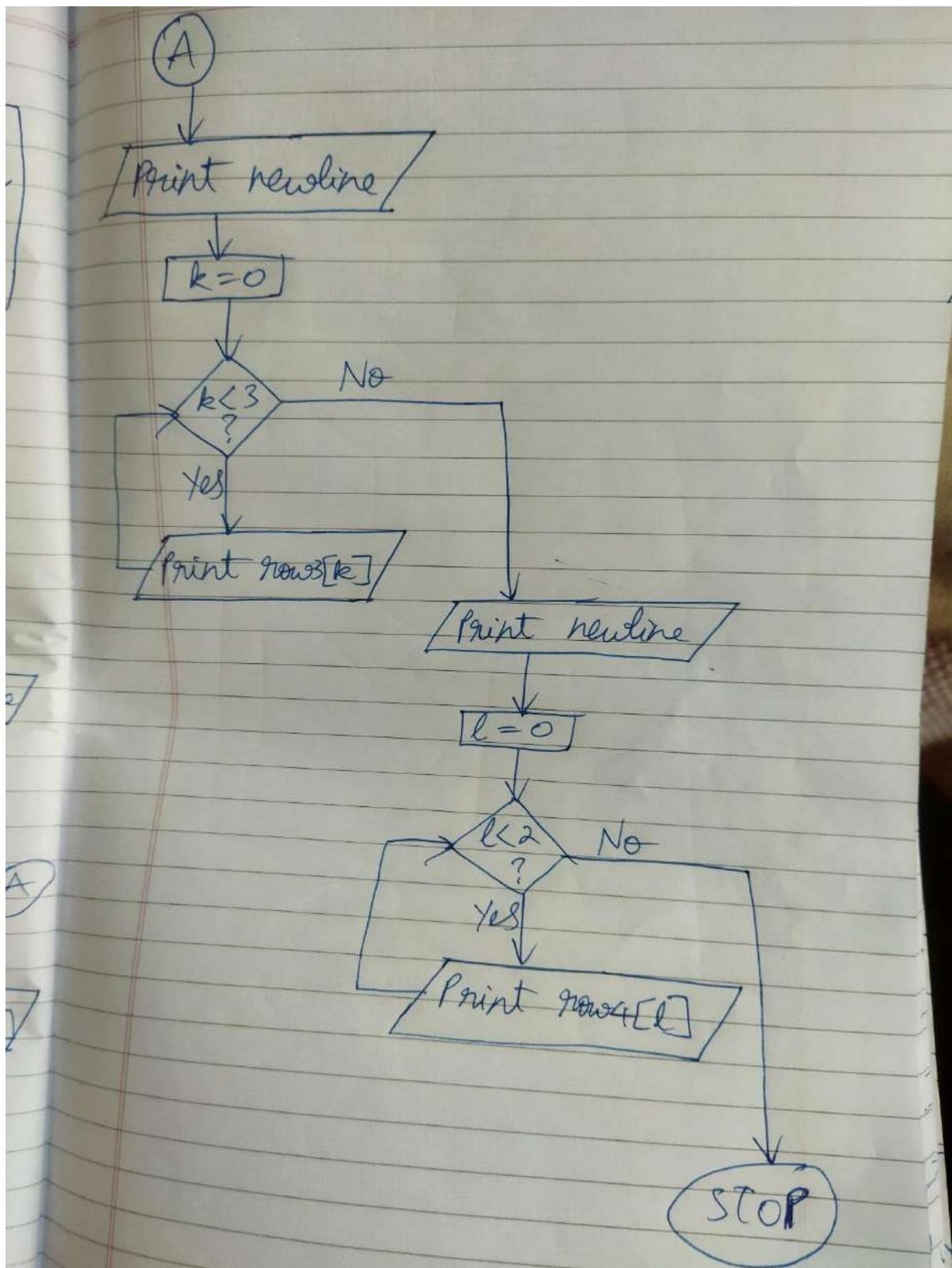
7 5 1 4 2
5 1 8 4
2 7 5
3 2

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- Arrays and for loop is used for this pyramid since the numbers making up the pyramid have no sequence.

d) 1. START
2. for i<=0 to 5 do
3. Print row1[i]
4. end for i<=5
5. Print "\n"
6. for j<=0 to 4 do
7. Print row2[j]
8. end for j<=4
9. Print "\n"
10. for k<=0 to 3 do
11. Print row3[k]
12. end for k<=3
13. Print "\n"
14. for l<=0 to 2 do
15. Print row4[l]
16. end for l<=2
17. STOP





5. Display Pascal's Triangle. Enter the number of rows: 6

```
#include <stdio.h>

int main() {
    int vline, coeff = 1, space, i, j;
    printf("Enter the number of rows (vertical lines): ");
    scanf("%d", &vline);

    for (i = 0; i < vline; i++) {
        for (space = 1; space <= vline - i; space++)
            printf(" ");
        for (j = 0; j <= i; j++) {
            if (j == 0 || i == 0)
                coeff = 1;
            else
                coeff = coeff * (i - j + 1) / j;
            printf("%4d", coeff);
        }
        printf("\n");
    }
    return 0;
}
```

The screenshot shows a C IDE interface with the following details:

- Toolbar:** Includes icons for Run, Debug, Stop, Share, and Save.
- File List:** Shows "main.c".
- Code Editor:** Displays the following C code:

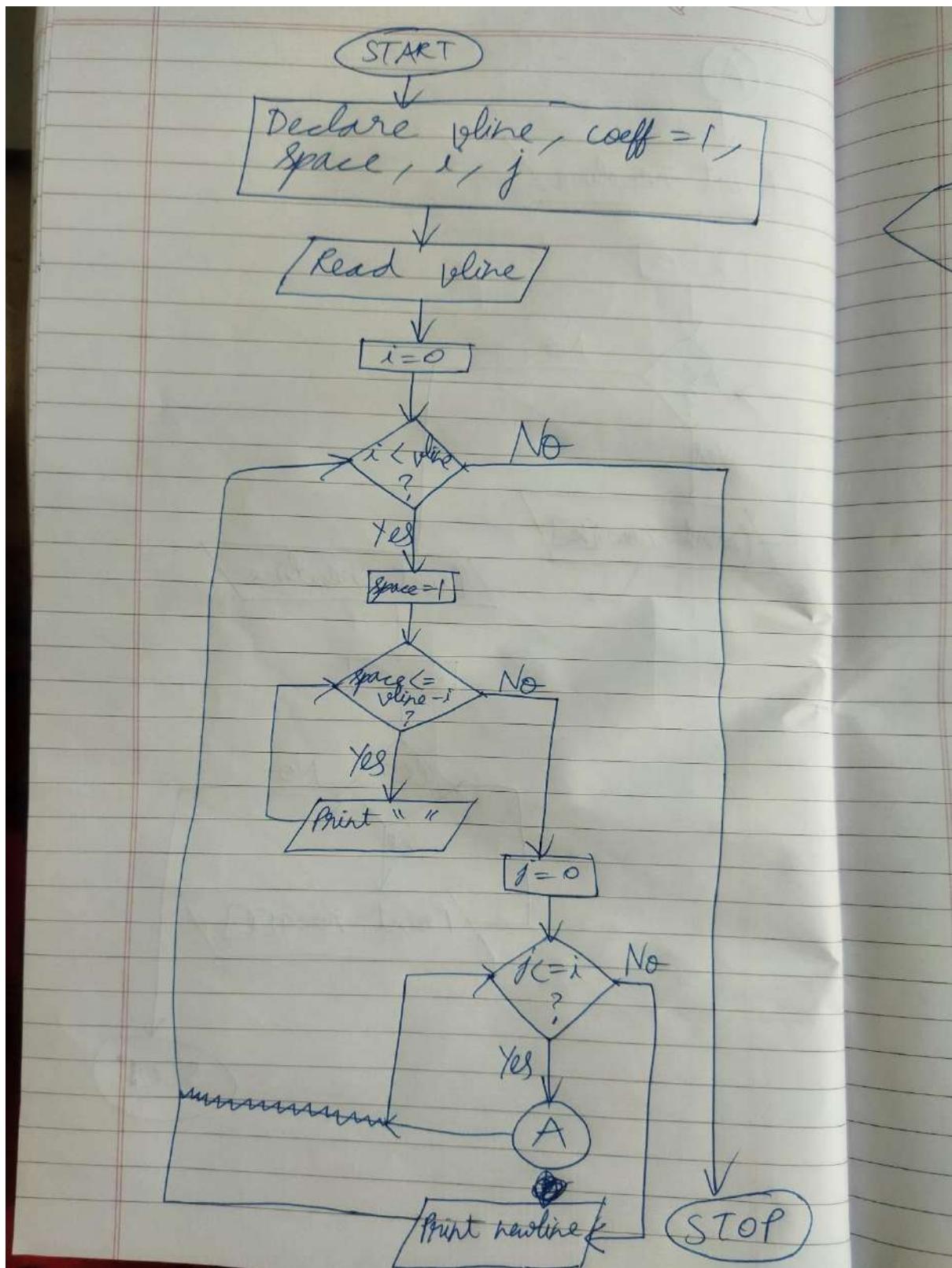
```
#include <stdio.h>
int main() {
    int vline, coeff = 1, space, i, j;
    printf("Enter the number of rows (vertical lines): ");
    scanf("%d", &vline);
    for (i = 0; i < vline; i++) {
        for (j = 0; j < i; j++)
            printf("   ");
        for (j = 0; j < coeff; j++)
            printf("%d ", coeff);
        coeff++;
        printf("\n");
    }
}
```
- Output Console:** Shows the program's output:

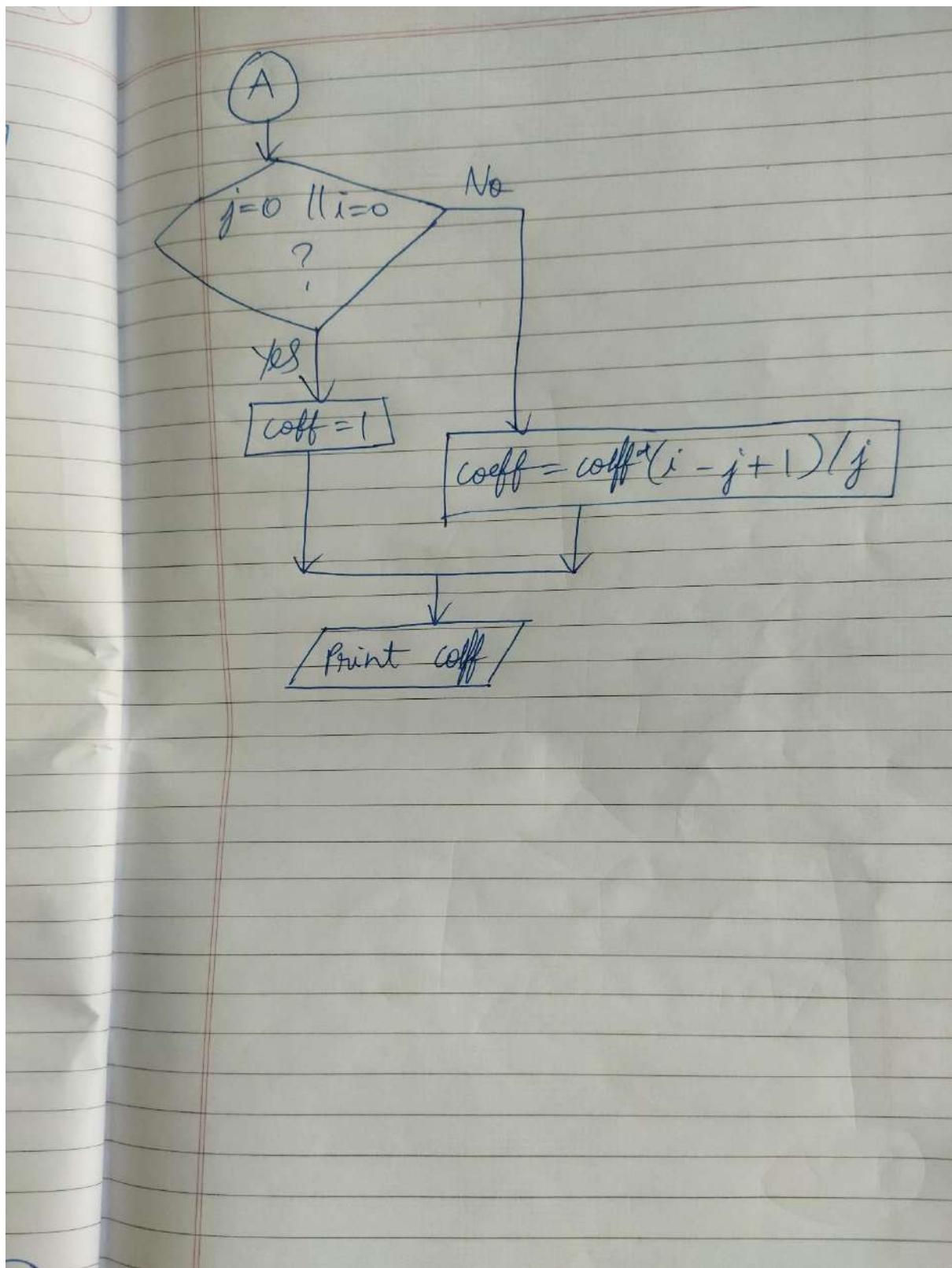
```
Enter the number of rows (vertical lines): 6
      1
      1   1
      1   2   1
      1   3   3   1
      1   4   6   4   1
      1   5   10  10  5   1

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- the variable called space is used to arrange the numbers in a pyramid-like equilateral triangle instead of a normal triangle. Variable vline tells number of rows in Pascal triangle to be printed. Variables i and j code for constituent numbers in rows and columns of Pascal triangle respectively.

e) 1. START
2. Read vline
3. for $i < 0$ to vline do
4. for space < 1 to vline - 1 do
5. Print ""
6. end for space $>$ vline - 1
7. for j < 0 to i do
8. If $j < 0$ OR $i < 0$
9. coeff $\leftarrow 1$
10. Else
11. coeff $\leftarrow coeff \times (i - j + 1) / j$
12. Print coeff
13. end for $j > i$
14. Print "n"
15. end for $i < vline$
16. STOP





LAB 4- Take Home

1. Write a program to find if an input character is a digit ,a lower case character, an upper case character or a special character.

```
#include<stdio.h>

int main()
{
    char ch;
    printf("Enter Any Character :");
    scanf("%c",&ch);

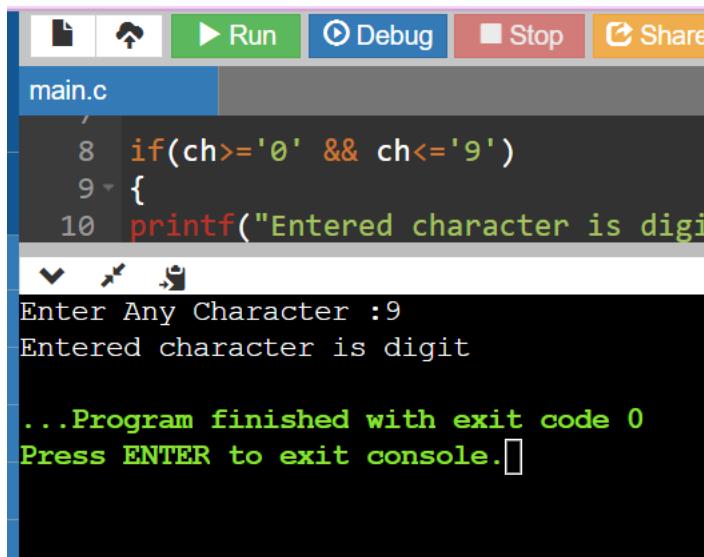
    if(ch>='0' && ch<='9')
    {
        printf("Entered character is digit");
    }
    else if(ch>='A' && ch<='Z')
    {
        printf("Entered character is uppercase letter");
    }
    else if(ch>='a' && ch<='z')
    {
        printf("Entered character is lowercase letter");
    }
    else
    {
        Samuel Abigail Mathew
        71762108039
    }
}
```

```
    printf("Entered character is special character");

}

return 0;

}
```



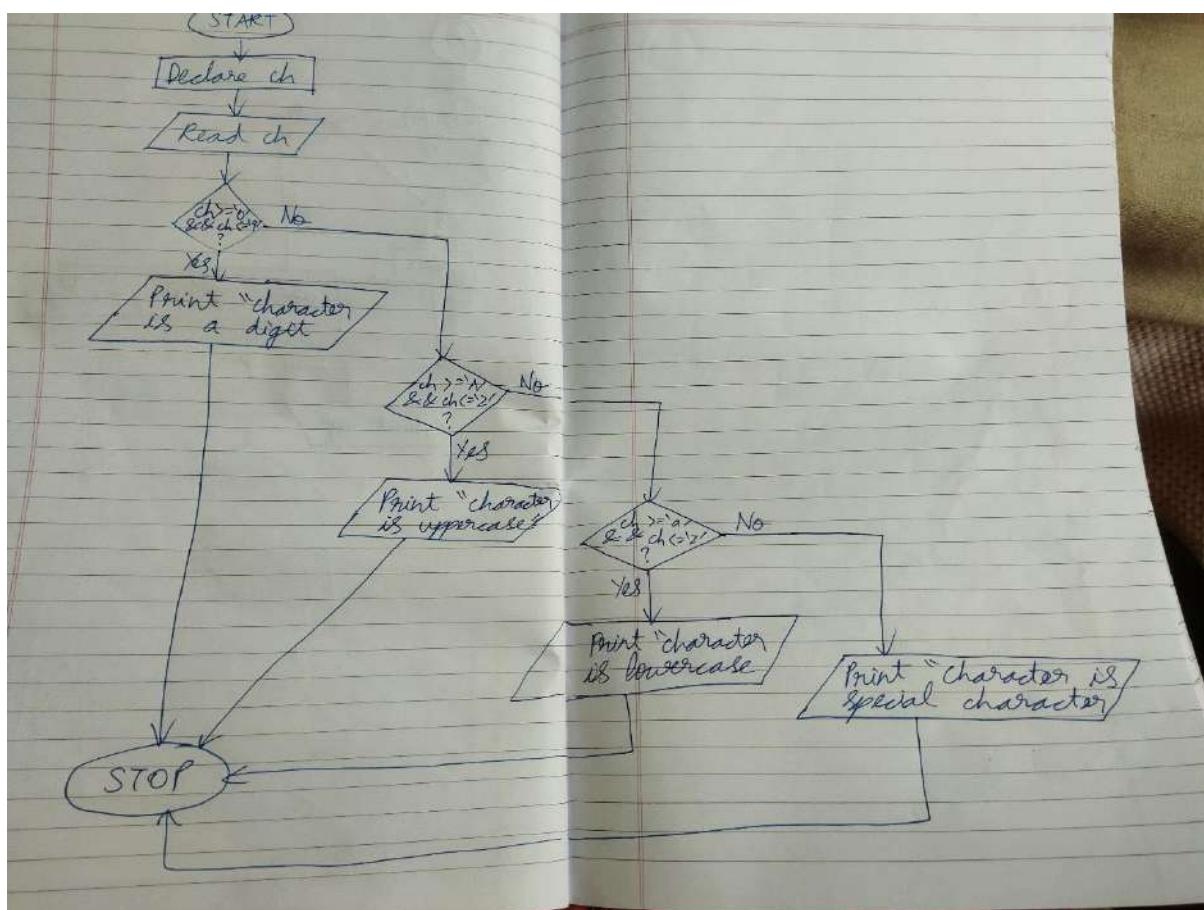
```
main.c
8 if(ch>='0' && ch<='9')
9 {
10 printf("Entered character is digit

Enter Any Character :9
Entered character is digit

...Program finished with exit code 0
Press ENTER to exit console.[]
```

Inference- if negative integers are typed as the character, then output will come as special character instead of digit since – (minus) is a special character. It works fine for positive integers though.

- 1) 1. START
2. Read ch
3. If $ch \geq '0'$ AND $ch \leq '9'$
4. Print "Entered character is digit"
5. Else if $ch \geq 'A'$ AND $ch \leq 'Z'$
6. Print "Entered character is uppercase letter"
7. Else if $ch \geq 'a'$ AND $ch \leq 'z'$
8. Print "Entered character is lowercase letter"
9. Else
10. Print "Entered character is special character"
11. STOP



2. Write a program which reads an integer n, and finds the value of constant e using the following series truncated to n terms: $1/e = 1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \dots$

```
#include <stdio.h>

int main()
{
    int i=1, j, n, fact, sign = 1;
    float e, sum=1, term;

    printf("Enter the value of n: ");
    scanf("%d", &n);

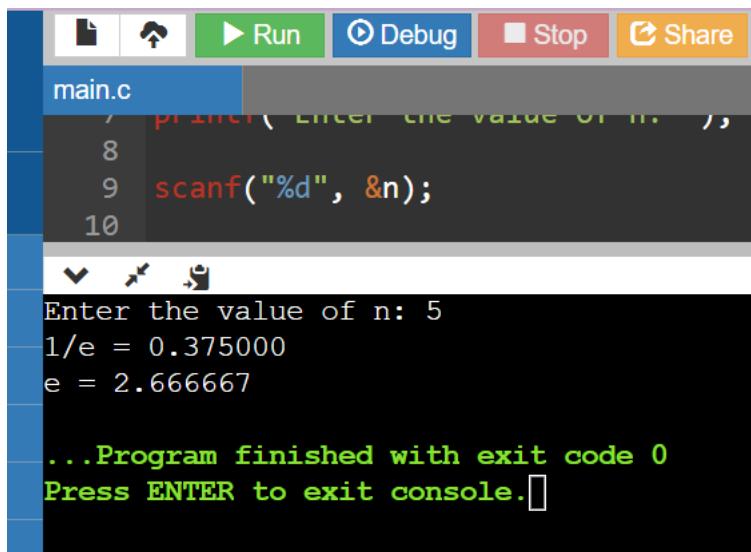
    for (i = 1; i <= (n-1); i++)
    {
        fact = 1;
        for (j = 1; j <= i; j++)
        {
            fact = fact * j;
        }

        sign = - 1 * sign;
        sum += sign * 1/(float)fact;
    }
}
```

Samuela Abigail Mathew
71762108039

```
printf("1/e = %f\n", sum);
printf("e = %f", 1/sum);

return 0;
}
```



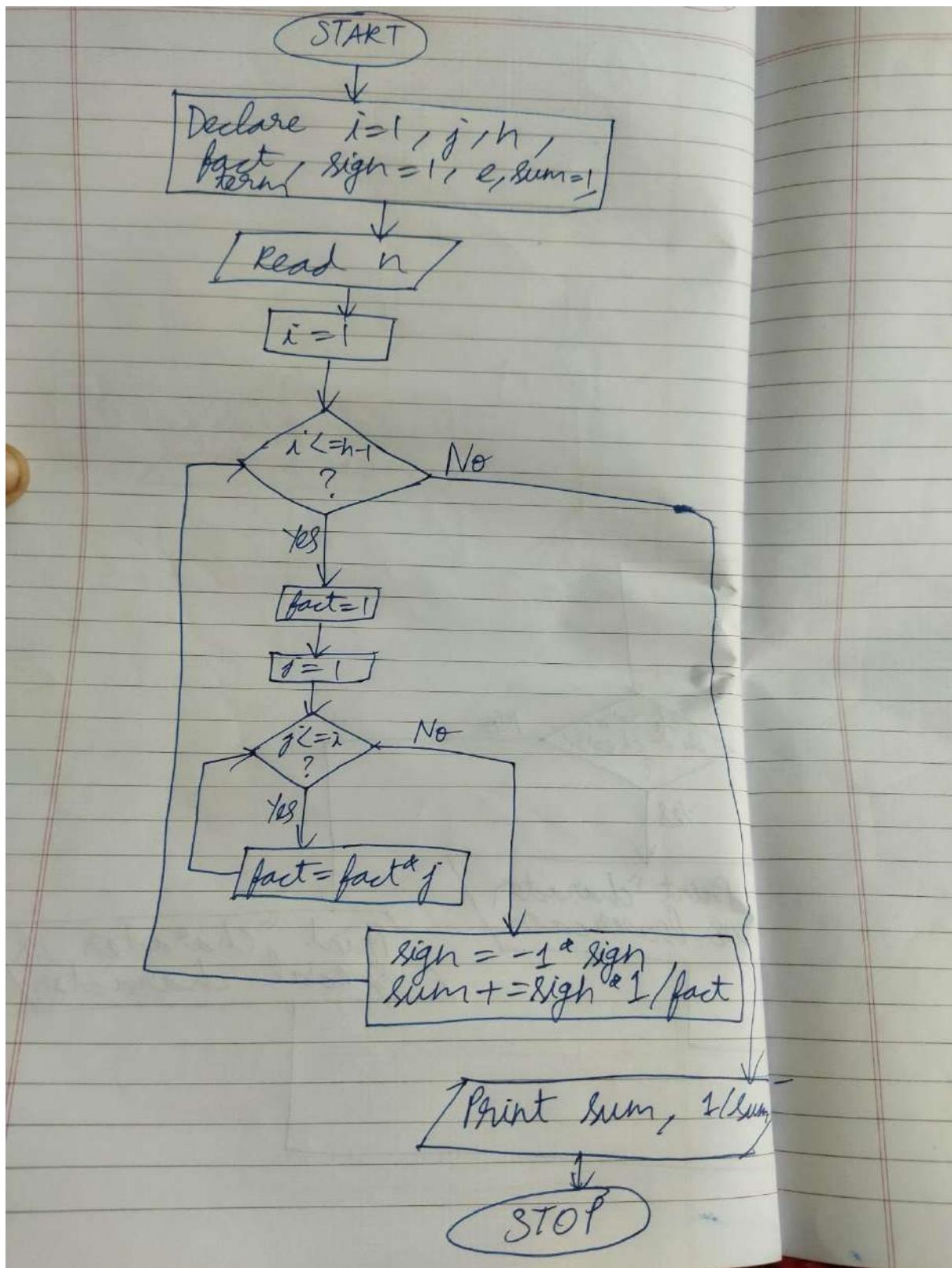
```
main.c
1/   printf( "Enter the value of n. " );
2/   8
3/   9   scanf( "%d", &n );
4/   10

Enter the value of n: 5
1/e = 0.375000
e = 2.666667

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- Value of e is 2.71828 approximately. The program coded above gives an e value which is closer to 2.71828 for all possible values of n.

2) 1. START
2. Read n
3. for $i \leftarrow 1$ to $n-1$ do
4. fact $\leftarrow 1$
5. for $j \leftarrow 1$ to i do
6. fact \leftarrow fact * j
7. end for $j \leftarrow i+1$
8. sign $\leftarrow 1 * sign$
9. sum $\leftarrow sum + sign * 1 / fact$
10. end for $i \leftarrow n$
11. Print sum
12. Print $1 / sum$
13. STOP



3. given two character arrays s1[25] and s2[25], check whether s2 is rotated string of s1 or not.

Ex: S1= tea; S2= eat; ---- Yes

S1 = Apple S2 = leap; ---- Yes

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

```
#include <string.h>
```

```
int main()
```

```
{
```

```
char s1[25], s2[25];
```

```
printf("Enter string 1:");
```

```
scanf("%s", s1);
```

```
printf("Enter string 2:");
```

```
scanf("%s", s2);
```

```
if(strlen(s1) != strlen(s2)){
```

```
printf("No");
```

```
return -1;
```

```
}
```

```
else{
```

```
//Concatenate string 1 with string 1 and store it in string 1
```

```
strcat(s1, s2);
```

```
//Check whether string 2 is present in string 1
```

Samuela Abigail Mathew

71762108039

```
if(strstr(s1, s2) != NULL)
printf("Yes");
else
printf("No");
}

return 0;
}
```

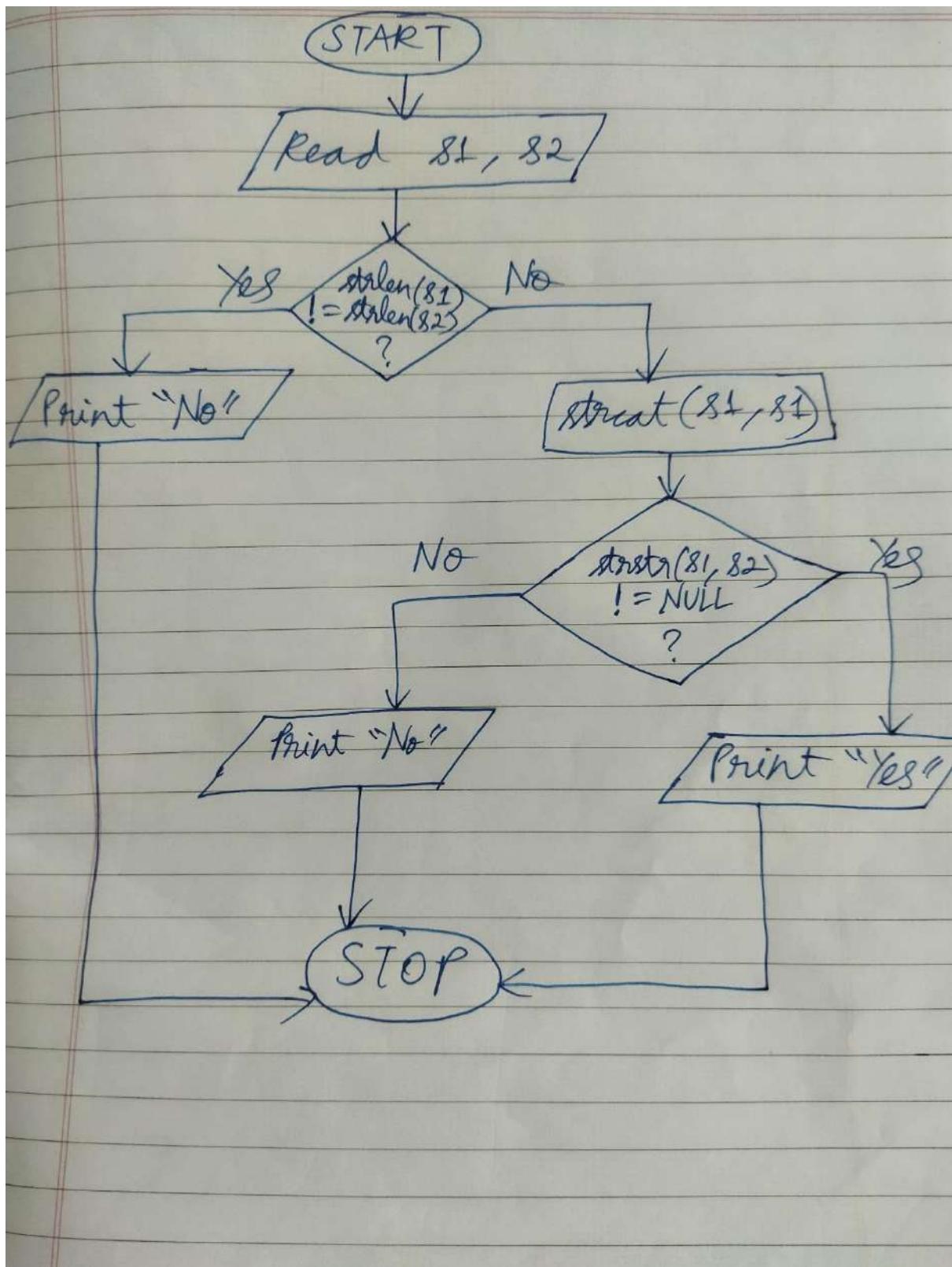
```
main.c
1 #include <stdio.h>
2 #include <string.h>
3
4 int main()
5 {
6     char s1[50], s2[50];
7     int i, j;
8
9     printf("Enter string 1:");
10    gets(s1);
11    printf("Enter string 2:");
12    gets(s2);
13
14    for(i = 0; s1[i] != '\0'; i++)
15    {
16        for(j = 0; s2[j] != '\0'; j++)
17        {
18            if(s1[i] == s2[j])
19            {
20                printf("%c", s1[i]);
21                break;
22            }
23        }
24    }
25
26    return 0;
27 }
```

```
Enter string 1:apple
Enter string 2:pleap
Yes

...Program finished with exit code 0
Press ENTER to exit console.█
```

Inference- string header file is used to code for this program.

- 3) 1. START
2. Read s_1, s_2
3. If $\text{strlen}(s_1) \neq \text{strlen}(s_2)$
4. Print "No"
5. Else
6. $\text{strcat}(s_1, s_2)$
7. If $\text{strstr}(s_1, s_2) \neq \text{NULL}$
8. Print "Yes"
9. Else
10. Print "No"
11. STOP



4. Evaluate a given expression which is represented as a character array. Ex: 4+120-8 Output: 116

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

int main()
{
    int i, k=0, length, expsum=0;
    char exptext[20], tempexp[20]="", op='+';

    //Take input of the expression.
    printf("\n");
    printf("Enter a Expression :");
    gets(exptext);

    length = strlen(exptext);//length of the expression string

    for (i=0; i<=length-1; i++)
    {
        printf("character of expression string at %d is %c\n", i, exptext[i]);

        if ((exptext[i]== '+') || (exptext[i]=='-'))
        {
            // Based on the operand expression adds or subtracts to the total
            sum
        }
    }
}
```

```
if (op == '+') expsum += atoi(tempexp);
if (op == '-') expsum -= atoi(tempexp);

k=0; // Reset temp array index controller
memset(tempexp, 0, 20); // Reset temp character array using
memset function
op = exptext[i];// Save the operand for next operation
}
else
{
tempexp[k] = exptext[i];
k=k+1;
}
}

// Calculation of the last part of the expression
if (op == '+') expsum += atoi(tempexp);
if (op == '-') expsum -= atoi(tempexp);

printf("Result of the Expression %s is %d\n", exptext, expsum);
return 0;
}
```

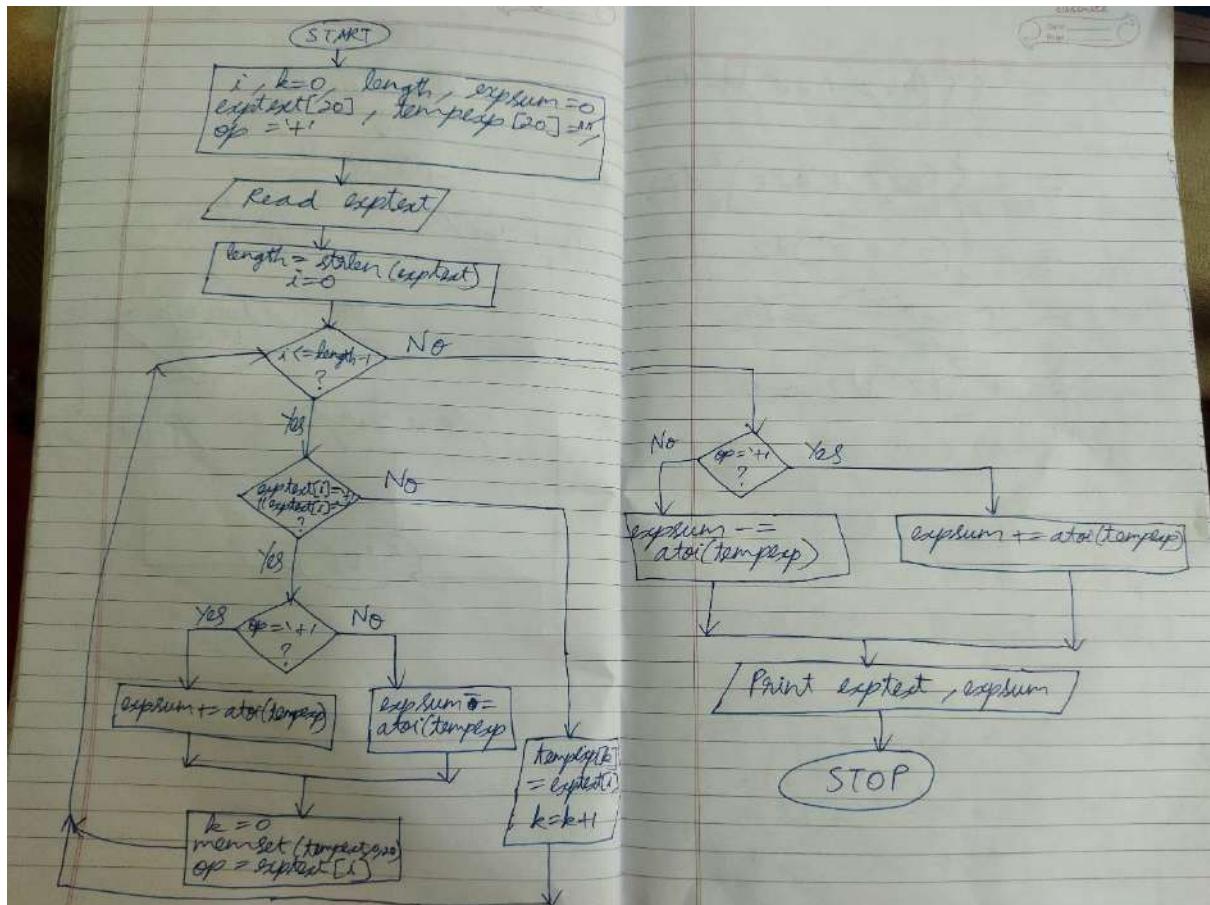
```
Enter a Expression :144+3-98
character of expression string at 0 is 1
character of expression string at 1 is 4
character of expression string at 2 is 4
character of expression string at 3 is +
character of expression string at 4 is 3
character of expression string at 5 is -
character of expression string at 6 is 9
character of expression string at 7 is 8
Result of the Expression 144+3-98 is 49
```

```
ct
```

```
...Program finished with exit code 0
Press ENTER to exit console.[]
```

Inference- it works for + and – operands only. It won't work if the expression has more than 3 terms. The numerical terms of the expression are stored in an array, and the expression as a whole acts as a string of characters.

- 4) 1. START
2. Read exptext
3. length \leftarrow strlen(exptext)
4. for $i \leftarrow 0$ to length - 1 do
5. Print i, exptext[i]
6. If exptext[i] $< '+'$ OR ~~exptext[i]~~
exptext[i] $< '-'$
7. If op $< '+'$.
8. expsum \leftarrow expsum + atoi(tempexp)
9. If op $< '-'$,
10. expsum \leftarrow expsum - atoi(tempexp)
11. k $\leftarrow 0$
12. memset(tempexp, 0, 20);
13. op \leftarrow exptext[i]
14. Else
15. tempexp[k] \leftarrow exptext[i]
16. k $\leftarrow k + 1$
17. end for i \leftarrow length
18. If op $< '+'$,
19. expsum \leftarrow expsum + atoi(tempexp)
20. If op $< '-'$,
21. expsum \leftarrow expsum - atoi(tempexp)
22. Print exptext, expsum
23. STOP



LAB 5

Aim- Learn to use strings.

Implement C programs for the following problem statements:

a) Palindromized Number: A string is said to be a palindrome if the order of arrangement of alphabet symbols read from both right to left ad left to right are one and the same. Similarly, let us claim that any number is said to be a palindromized number if the order of arrangement of digits from left to right and right to left are same.

Given a number 'x' check whether it is a palidromized number or not.

Ex 2: x = 12321 output : YES

Ex 3: x = 12211 output : NO

Ex 4: x = 8 output : YES

```
#include <stdio.h>

int main() {
    int n, temprev= 0, r, original;
    printf("Enter an integer: ");
    scanf("%d", &n);
    original=n;
```

```
// reversed integer is stored in another variable  
while (n != 0) {  
    r = n % 10;  
    temprev = temprev * 10 + r;  
    n /= 10;  
}  
  
//checks if palindrome or not  
if (original == temprev)  
    printf("Reverse of %d is %d , so it is a palindrome.",original,  
temprev);  
else  
    printf("Reverse of %d is %d , so it is not a  
palindrome.",original,  
temprev);  
return 0;  
}
```

The screenshot shows a C IDE interface with the following details:

- Toolbar:** Includes icons for file operations (New, Open, Save), Run, Debug, Stop, Share, and Beautify.
- Code Editor:** A tab labeled "main.c" contains the following C code:

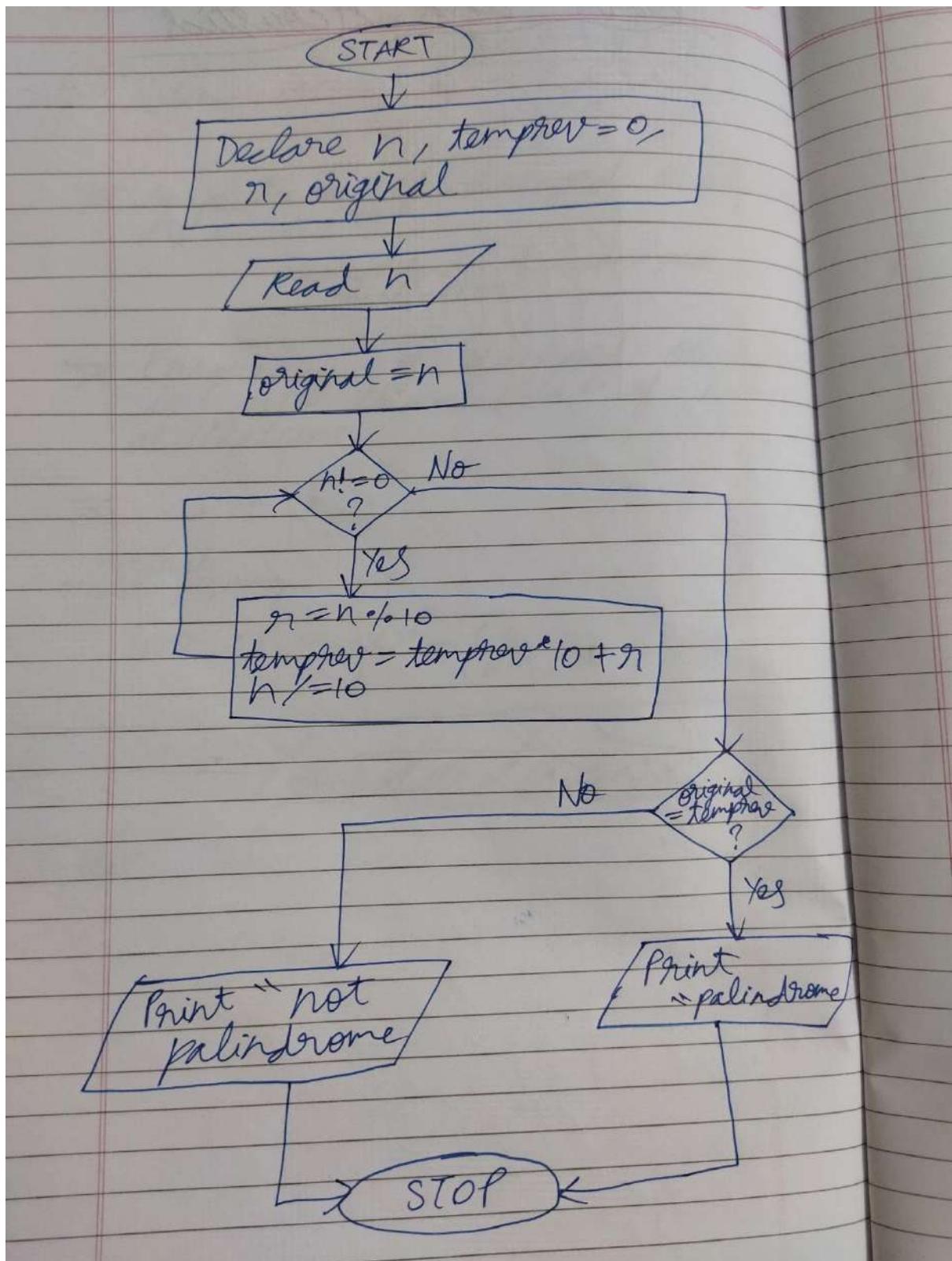
```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Enter an integer: ");
6     scanf("%d", &n);
7     original=n;
```
- Output Console:** Displays the following output:

```
Enter an integer: 4865
Reverse of 4865 is 5684 , so it is not a palindrome.

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- n is the variable which initially stores the entered number, temprev is the variable storing reversed number, and original is the variable storing the entered number. The program checks if temprev is same as original to declare the entered number as palindrome. If not, then the entered number is declared as not palindrome.

- D) 1. START
2. Read n
3. original $\leftarrow n$
4. while $n! = 0$
5. $r \leftarrow n \% 10$
6. temprev $\leftarrow temprev * 10 + r$
7. $n \leftarrow n / 10$
8. If original \leftarrow temprev
9. Print "It is palindrome"
10. Else
11. Print ("It is not palindrome")
12. STOP



b) Anagrams: Two strings are said to be Anagrams if they possess same number of individual alphabet symbols and

they differ by order of arrangements. For example, the strings ‘dog’ and ‘god’ are Anagrams whereas ‘good’ and ‘dog’ are not Anagrams. Given two strings S1 and S2, check whether they are Anagram strings are not.

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

```
#include <string.h>
```

```
int main () {
```

```
    char s1[25], s2[25], tempstr;
```

```
    int i, j, n, m;
```

```
    printf("Enter string 1:");
```

```
    scanf("%s", s1);
```

```
    printf("Enter string 2:");
```

```
    scanf("%s", s2);
```

```
    n = strlen(s1);
```

```
    m = strlen(s2);
```

```
// If both strings are of different length, then they are not  
anagrams
```

```
if( n != m) {  
    printf("Strings are not anagrams \n");  
    return 0;  
}  
  
  
for (i = 0; i < n-1; i++) {  
    for (j = i+1; j < n; j++) {  
        if (s1[i] > s1[j]) {  
            tempstr = s1[i];  
            s1[i] = s1[j];  
            s1[j] = tempstr;  
  
        }  
    }  
}  
  
if (s2[i] > s2[j]) {  
    tempstr = s2[i];  
    s2[i] = s2[j];  
    s2[j] = tempstr;  
}  
}  
}  
}
```

// Compare both strings character by character

Samuela Abigail Mathew
71762108039

```

for(i = 0; i<n; i++) {
    if(s1[i] != s2[i]) {
        printf("Strings are not anagrams \n");
        return 0;
    }
}

printf("Strings are anagrams \n");
return 0;
}

```

The screenshot shows a code editor interface with a toolbar at the top and a code editor area below. The code editor shows a file named 'main.c' with the following content:

```

main.c
13
14 n = strlen(s1);
15 m = strlen(s2);

```

Below the code editor is a terminal window displaying the program's output:

```

Enter string 1:god
Enter string 2:gdo
Strings are anagrams

...Program finished with exit code 0
Press ENTER to exit console.

```

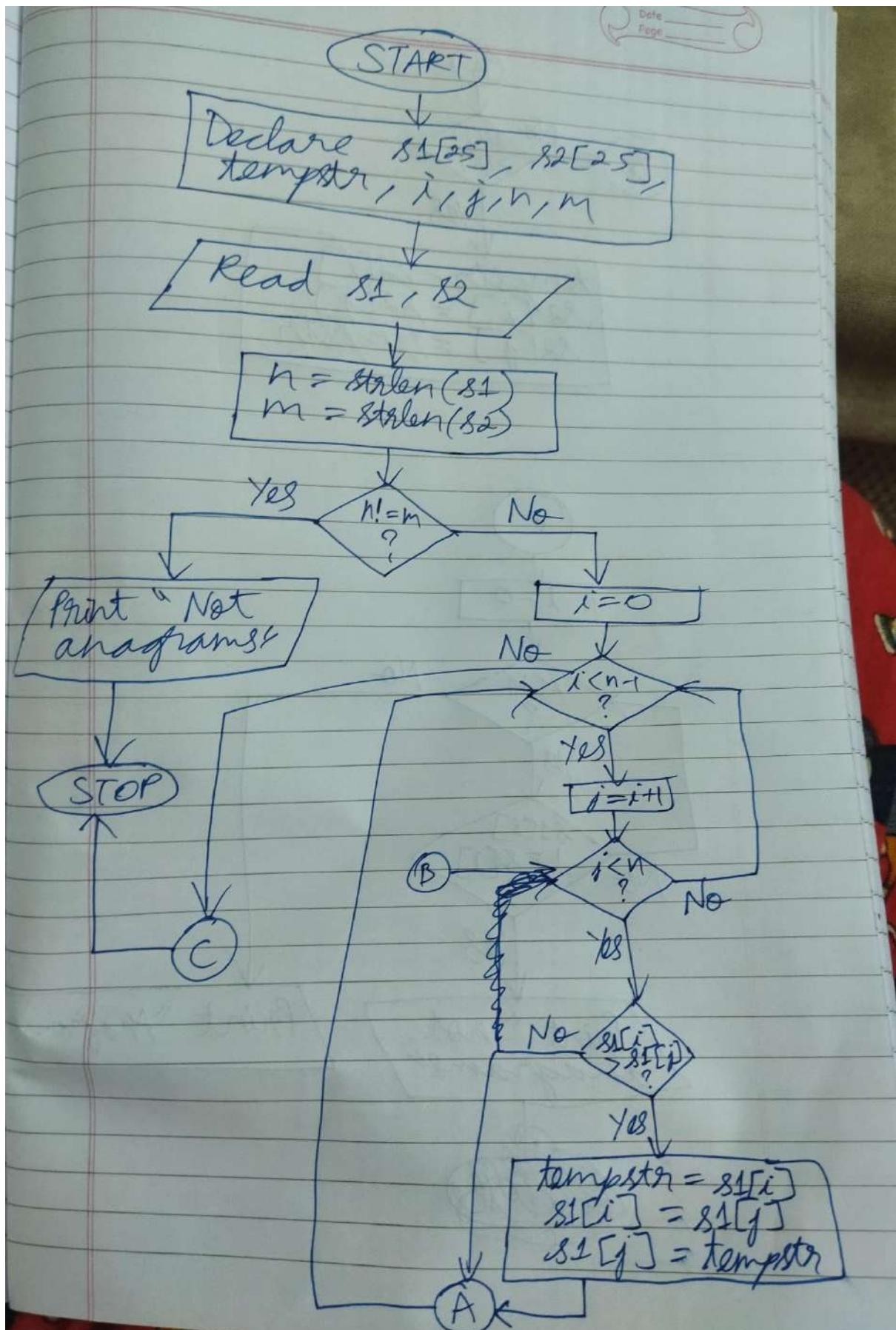
Inference- two strings are taken as input and then they are checked for string length. If both have same length then the program further checks by arranging the characters of the string in ascending order and checking each character of the

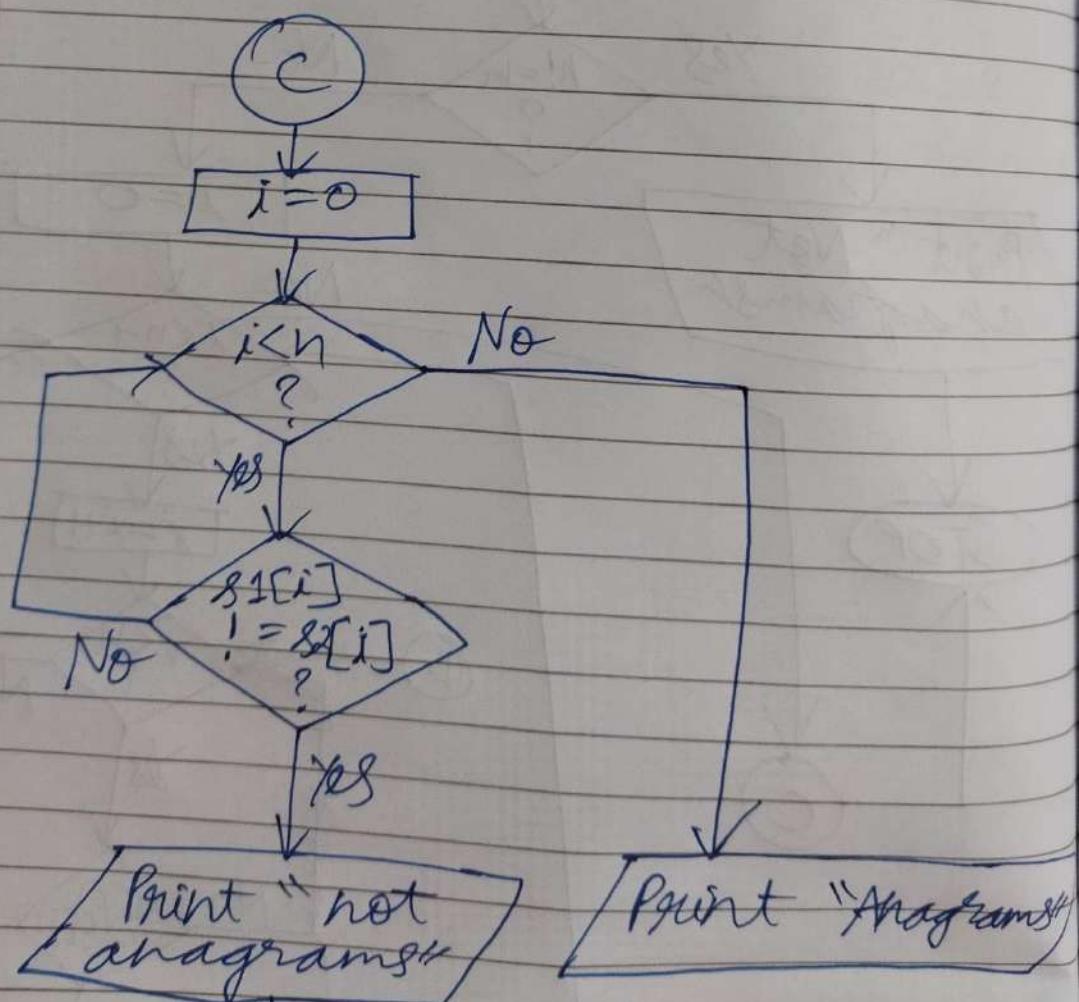
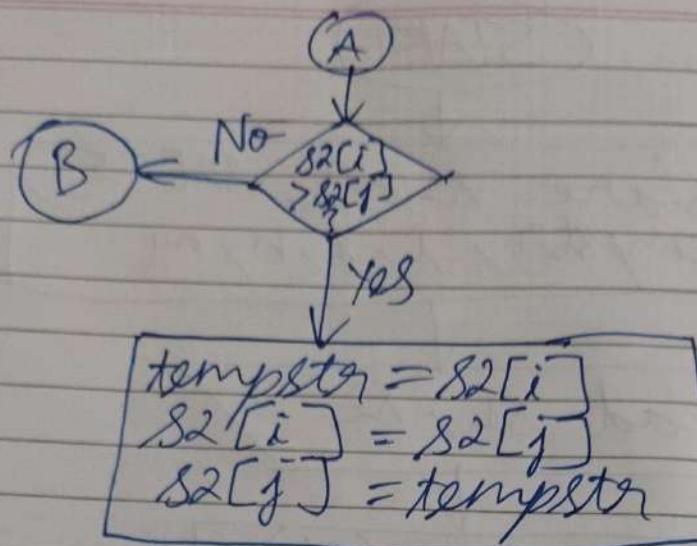
string. This is how the program checks if the strings are anagrams or not.

2) 1. START
2. Read s_1 s_2
3. $n \leftarrow \text{strlen}(s_1)$
4. $m \leftarrow \text{strlen}(s_2)$
5. If $n \neq m$
6. Print "Not anagrams"
7. Exit
8. for $i \leftarrow 0$ to $n-1$ do
9. for $j \leftarrow i+1$ to n do
10. If $s_1[i] > s_1[j]$
11. $\text{tempstr} \leftarrow s_1[i]$
12. $s_1[i] \leftarrow s_1[j]$
13. $s_1[j] \leftarrow \text{tempstr}$
14. If $s_2[i] > s_2[j]$
15. $\text{tempstr} \leftarrow s_2[i]$
16. $s_2[i] \leftarrow s_2[j]$
17. $s_2[j] \leftarrow \text{tempstr}$
18. end for $j \leftarrow n$
19. end for $i \leftarrow n-1$
20. for $i \leftarrow 0$ to n do

and is not palindrom

21. If $s1[i] \neq s2[i]$
22. Print ("Not anagrams")
23. Exit
24. end for $i \leq n$
25. Print "Anagrams"
26. STOP





LAB 6

Aim- Learn to match patterns in strings.

Given a base string and a pattern, check whether the pattern has occurred in the base string or NOT. if occurred, print the index position of the base string from where the pattern is found

```
#include <stdio.h>
#include <string.h>
int match(char [], char []);
int main() {
    char a[100], b[100];
    int position;

    printf("Enter string: ");
    gets(a);

    printf("Enter string pattern to find: ");
    gets(b);

    position = match(a, b);
```

```
if (position != -1) {
    printf("Found at location: %d\n", position + 1);
}
else {
    printf("Not found.\n");
}

return 0;
}

int match(char text[], char pattern[]) {
    int c, d, e, text_length, pattern_length, position = -1;

    text_length = strlen(text);
    pattern_length = strlen(pattern);

    if (pattern_length > text_length) {
        return -1;
    }

    for (c = 0; c <= text_length - pattern_length; c++) {
        position = e = c;

        for (d = 0; d < pattern_length; d++) {
```

Samuela Abigail Mathew
71762108039

```
if (pattern[d] == text[e]) {  
    e++;  
}  
}  
  
else {  
    break;  
}  
}  
}  
}
```

```
if (d == pattern_length) {
```

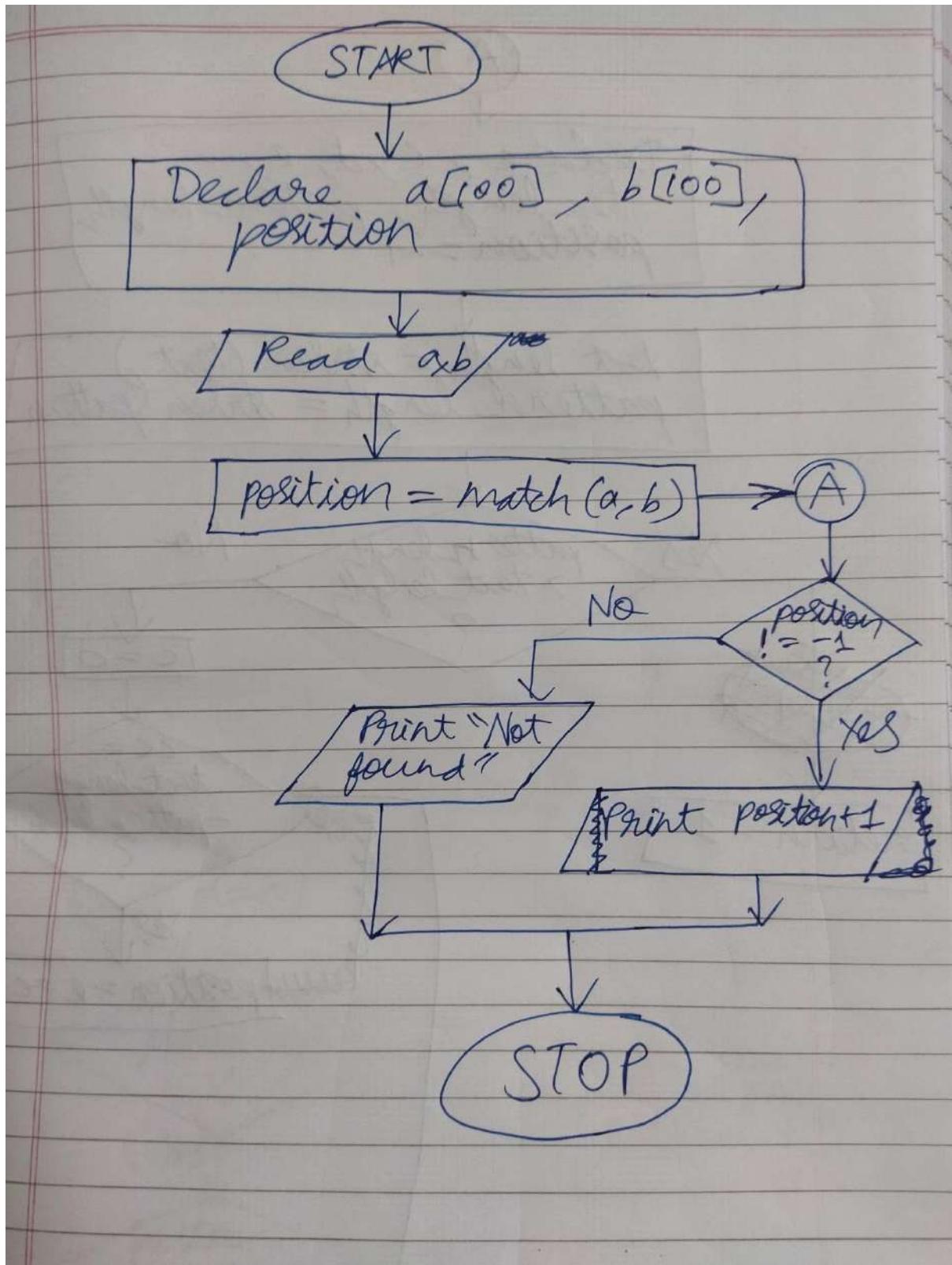
```
    return position;  
}  
}  
}  
  
return -1;  
}
```

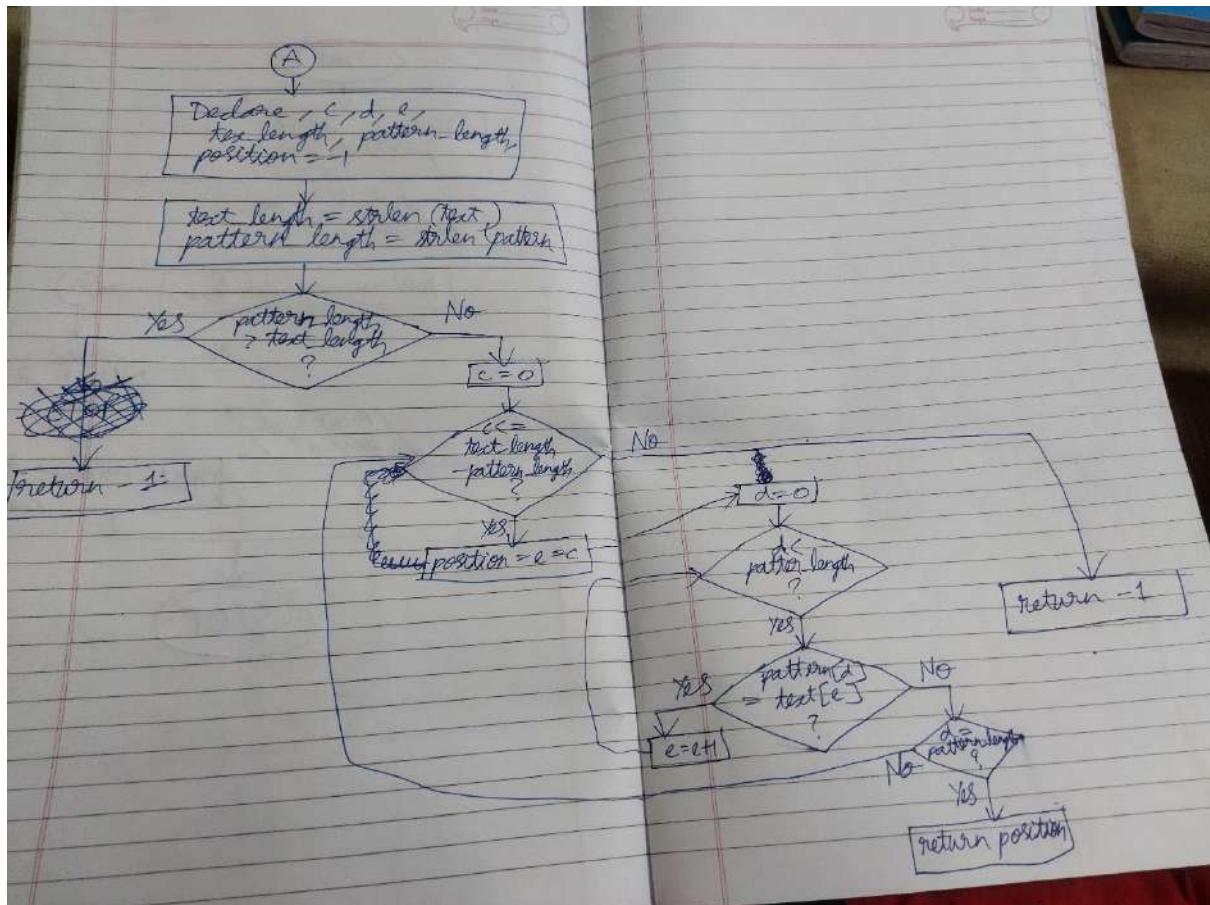
```
main.c
37 for (c = 0; c <= text_length - pattern_length; c++) {
38     position = e = c;
39
40
41                                     input
main.c:10:1: warning: 'gets' is deprecated [-Wdeprecated-declarations]
10 | gets(a);
| ^~~~
In file included from main.c:1:
/usr/include/stdio.h:577:14: note: declared here
577 | extern char *gets (char *__s) __wur __attribute_deprecated__;
| ^~~~
main.c:13:1: warning: 'gets' is deprecated [-Wdeprecated-declarations]
13 | gets(b);
| ^~~~
In file included from main.c:1:
/usr/include/stdio.h:577:14: note: declared here
577 | extern char *gets (char *__s) __wur __attribute_deprecated__;
| ^~~~
/usr/bin/ld: /tmp/ccTv021K.o: in function `main':
main.c:(.text+0x3a): warning: the `gets' function is dangerous and should not be used.
Enter string: mintcookie
Enter string pattern to find: co
Found at location: 5

fact ...Program finished with exit code 0
Press ENTER to exit console.[]
```

Inference- We are basically finding and giving first position of substring in entered string.

1. START
2. Read a, b
3. position \leftarrow match(a, b)
4. text_length \leftarrow strlen(text)
5. pattern_length \leftarrow strlen(pattern)
6. If (pattern_length $>$ text_length)
7. return -1
8. for c \leftarrow 0 to text_length - pattern_length
do
9. position \leftarrow c \leftarrow c
10. for d \leftarrow 0 to pattern_length do
11. If pattern[d] \leftarrow text[e]
12. e \leftarrow e + 1
13. Else
14. break
15. end for d \leftarrow pattern_length
16. end for c \leftarrow text_length - pattern_length
17. If d \leftarrow pattern_length
18. return position
19. If position \neq -1
20. Print position + 1
21. Else
22. Print "Not found"
23. STOP





LAB 7

Aim- Learn to remove duplicates in strings.

Removing Duplicates: Given a string ‘str’, remove all adjacent duplicate alphabet symbols present in it by retaining one as the representative.

Ex 1: str = ‘aaaabbbbbcccccdeeeee’;

output : ‘abcde’

Ex 2: str = ‘xyzz’;

output : ‘xyz’

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

```
#include <string.h>
```

```
int main()
```

```
{
```

```
char str[30]= "beepboopbeep";
```

```
int length= strlen(str);
```

```
for(int i=0; i<length; i++)
```

```
{
```

```
char ch = str[i];
```

```
//Check if current character matches any other character in
the subsequence
for(int j=i+1; j<length; ){

if(str[i] == str[j]){

//If yes, then shift the right characters to the left
for(int k=j; k<length; k++){
str[k] = str[k+1];
}

length--;
} else {
//only increment if the duplicate is not found
//because after the shift, index j can again have duplicate
j++;
}

}

printf("%s",str);
return 0;
}
```

Samuela Abigail Mathew
71762108039

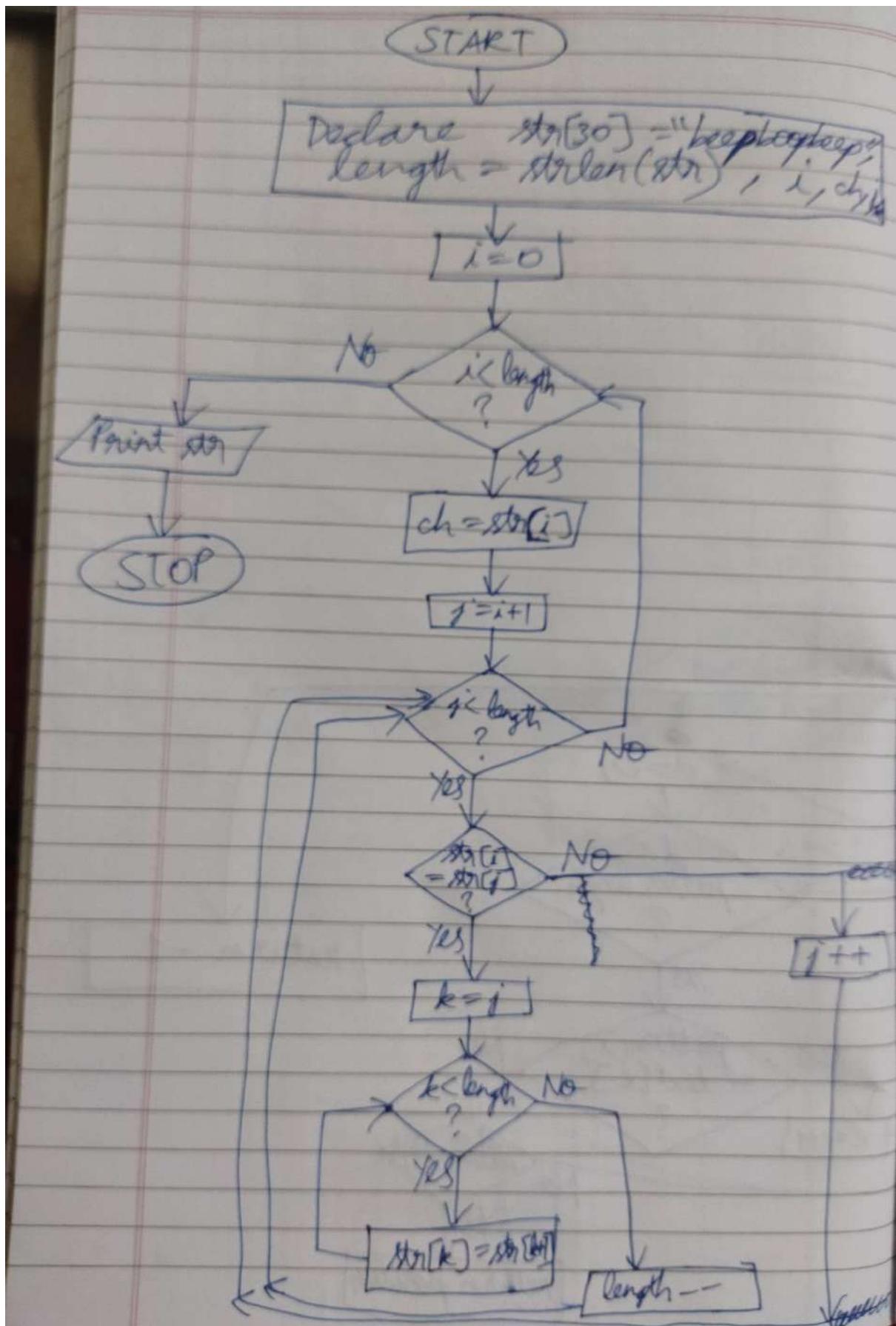
The screenshot shows a code editor interface with a dark theme. At the top, there is a toolbar with icons for file operations (Save, Open, Run, Debug, Stop, Share, Save, Beautify) and a download icon. The current file is "main.c". The code itself is as follows:

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int main()
5 {
6     char str[30] = "beepboopbeep";
7     int length = strlen(str);
8
9     for(int i=0; i<length; i++)
10    {
11        char ch = str[i];
12
13        //Check if current character matches any other character in
14        for(int j=i+1; j<length; j++){
15
16            if(str[i] == str[j]){
17
18                //If yes, then shift the right characters to the left
19                for(int k=j; k<length; k++){
20                    str[k] = str[k+1];
21                }
22            }
23        }
24    }
25
26    printf("%s", str);
27
28    return 0;
29}
```

In the bottom left, there is a terminal window showing the output of the program. The input was "bopo", and the output is "bopo... Program finished with exit code 0 Press ENTER to exit console. []".

Inference- Every time we find a duplicate character, we shift the position of other characters towards the left to remove duplicates, and we keep reducing length of string by 1 every time we do this.

1. START
2. Read str
3. length \leftarrow strlen(str)
4. for $i \leftarrow 0$ to length do
5. ch \leftarrow str[i]
6. for $j \leftarrow i+1$ to length do
7. If str[i] $<$ str[j]
8. for $k \leftarrow j$ to length do
9. str[k] \leftarrow str[k+1]
10. end for k \leftarrow length
11. length \leftarrow length - 1
12. Else
13. j \leftarrow j + 1
14. end for j \leftarrow length
15. end for i \leftarrow length
16. Print str
17. STOP



LAB 8

Aim- Learn to use concept of arrays.

1) Write a Program for deletion of an element from the specified location from Array.

```
#include <stdio.h>

int main()
{
    int array[100], position, c, n;

    printf("Enter number of elements in array\n");
    scanf("%d", &n);

    printf("Enter %d elements\n", n);

    for (c = 0; c < n; c++)
        scanf("%d", &array[c]);

    printf("Enter the location where you wish to delete
element\n");
```

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

if (position >= n+1)
printf("Deletion not possible.\n");
else
{
for (c = position - 1; c < n - 1; c++)
array[c] = array[c+1];

printf("Resultant array:\n");

for (c = 0; c < n - 1; c++)
printf("%d\n", array[c]);
}

return 0;
}
```

The screenshot shows a C IDE interface with the following details:

- Toolbar:** Includes icons for file operations (New, Open, Save), Run, Debug, Stop, Share, and Beautify.
- Code Editor:** The file "main.c" is open, containing the following code:

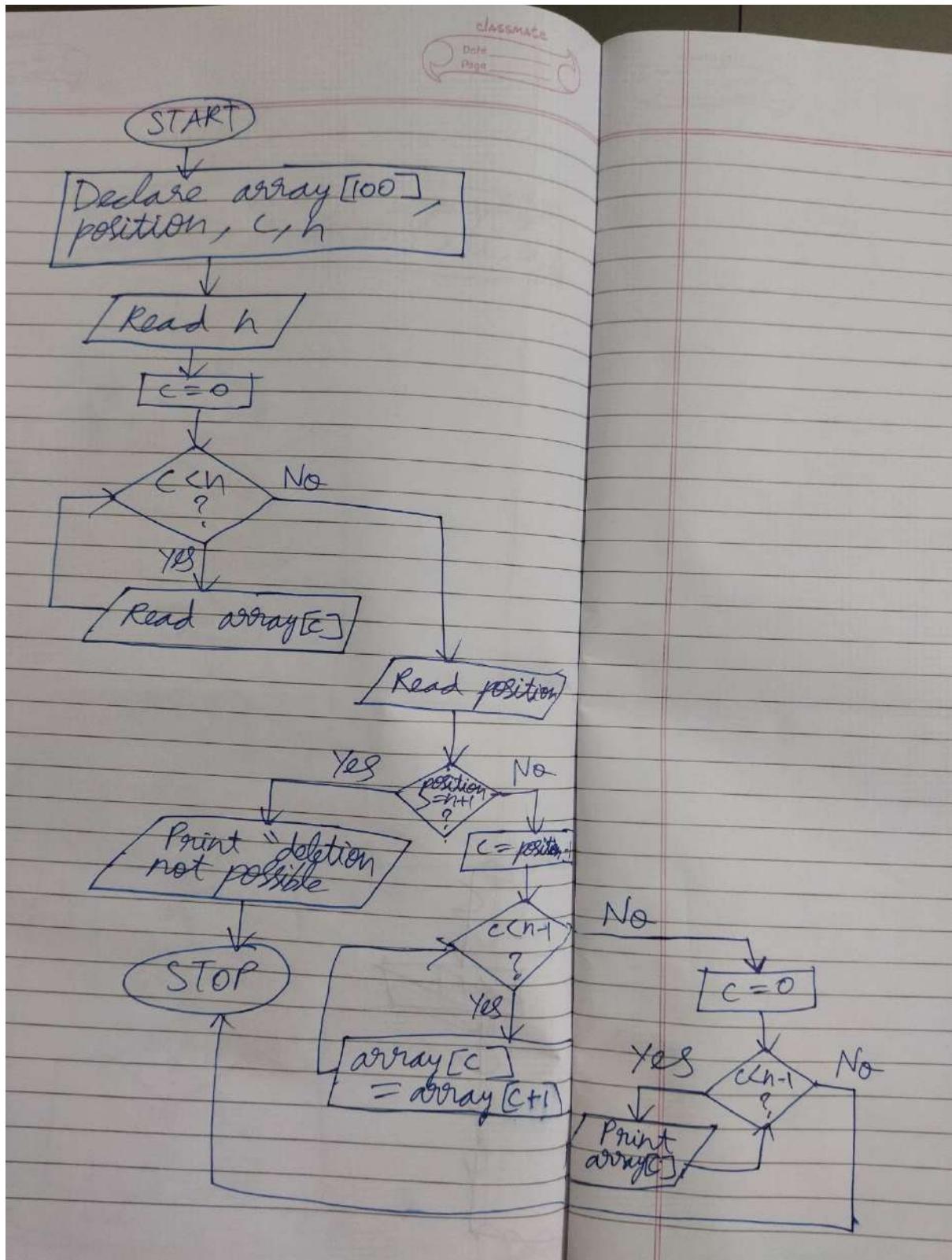
```
12  scanf("%d", &array[0]),  
13  
14  printf("Enter the location where you wish to delete ele  
15  scanf("%d", &position);
```
- Output Window:** Labeled "input", showing the program's interaction with the user:

```
Enter number of elements in array  
5  
Enter 5 elements  
45  
-89  
68  
3  
0  
Enter the location where you wish to delete element  
4  
Resultant array:  
45  
-89  
68  
0
```

...Program finished with exit code 0
Press ENTER to exit console. █

Inference- We are shifting position of all arrays elements which come after the entered position towards left to delete that particular element by overwriting it.

1) 1. START
2. Read n
3. for $c \leq 0$ to n do
4. Read array[c]
5. end for $c \leq n$
6. Read position
7. If position $\geq n+1$
8. Print "deletion not possible"
9. Else
10. for $c < position - 1$ to $n-1$ do
11. array[c] \leftarrow array[$c+1$]
12. end for $c < n-1$
13. for $c \leq 0$ to $n-1$ do
14. Print array[c]
15. end for $c < n-1$
16. STOP



2) Program to print unique elements in an array.

```
#include <stdio.h>
#include <stdlib.h>
int uniq(int arr[], int n){
    int i,j;
    for(i = 0; i < n; i++){
        for(j = 0; j < n; j++){
            if(arr[i] == arr[j] && i != j)
                break;
        }
        if(j == n ){
            printf("\nunique elements in the array: %d \n",arr[i]);
        }
    }
    return -1;
}
int main(){
    int n,i;
    printf("\nEnter number of elements : ");
    scanf("%d",&n);
    int arr[n];
    printf("\nEnter the array elements : ");
}
```

```

for(i = 0; i < n; i++){
    scanf("%d",&arr[i]);
}
uniq(arr, n);
return 0;
}

```

The screenshot shows a code editor interface with a toolbar at the top and a code editor window below. The code editor window displays a file named 'main.c' with the following content:

```

main.c
10 j
11 if(j == n ){
12 printf("\nunique elements in the array: %d \n",arr[i]);
13 ++count;

```

The terminal window below shows the execution of the program. It prompts for the number of elements (5), then asks for array elements (34, 8, 34, 0, 34). The output shows the count of unique elements (8) and the count of duplicates (0). The program exits with an exit code of 0.

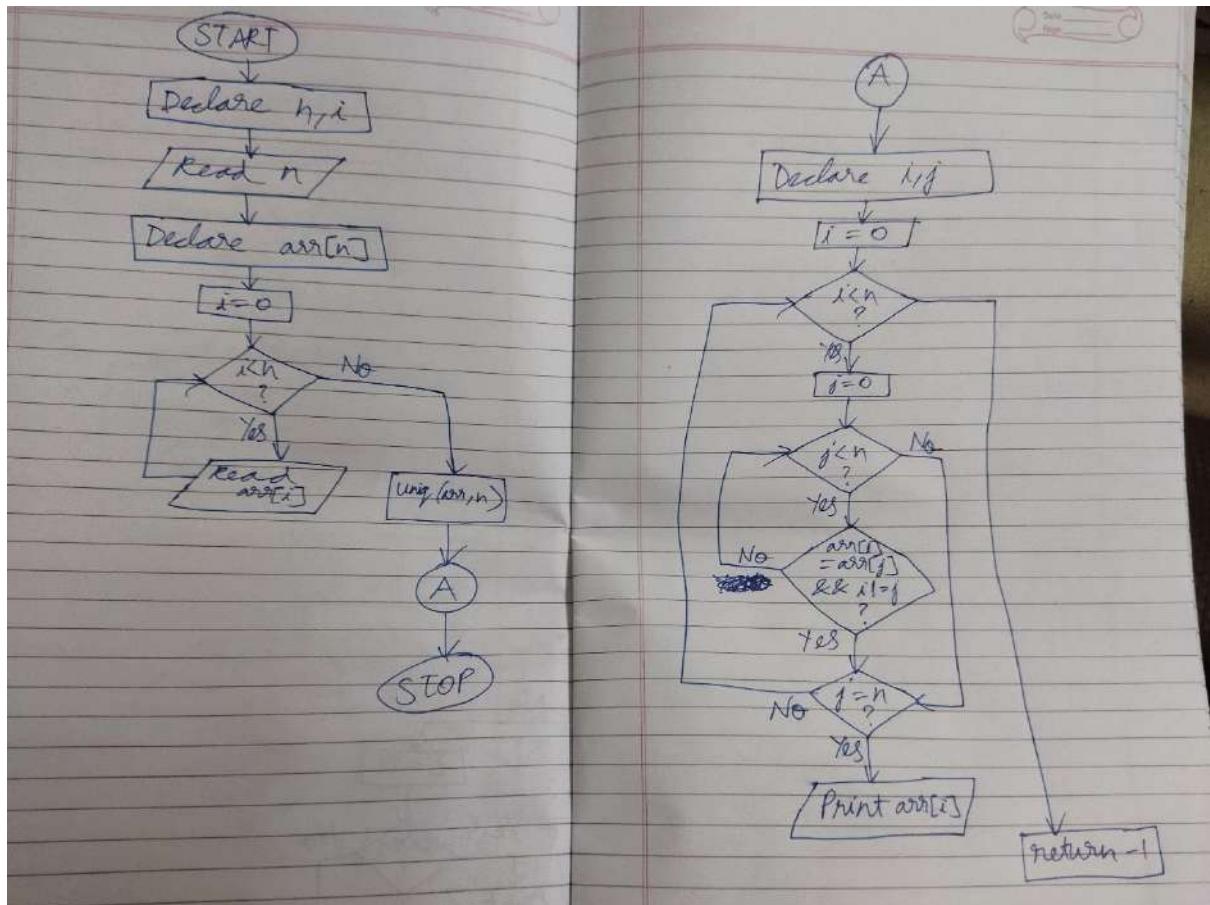
```

Enter number of elements : 5
enter the array elements : 34
8
34
0
34
unique elements in the array: 8
unique elements in the array: 0
...Program finished with exit code 0
Press ENTER to exit console. []

```

Inference- We are checking for all array elements for repetition, and if they are not repeated, they are printed as unique elements.

2) 1. START
2. Read n
3. for $i \leftarrow 0$ to n do
4. Read arr[i]
5. end for $i \leftarrow n$
6. call uniq(arr, n)
7. for $i \leftarrow 0$ to n do
8. for $j \leftarrow 0$ to n do
9. If $arr[i] < arr[j]$ AND $i \neq j$
10. break
11. end for $j \leftarrow n$
12. If $j \leq n$
13. Print arr[i]
14. end for $i \leftarrow n$
15. STOP



3) Find the Peak Element in an array and give the position

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

```
int main(){
    int arr[20], i=0, j=0, size, peak ;

    printf("\nEnter array size : ");
    scanf("%d", &size);

    for (i = 0; i < size; i++) {
```

Samuela Abigail Mathew
71762108039

```
printf("Enter arr[%d]: ", i);
scanf("%d", &arr[i]);
}

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

printf("\nPeak element of array is %d ", peak);
printf("\nPosition of peak is %d", j+1);
return 0;
}
```

The screenshot shows a C IDE interface with a toolbar at the top and a code editor window below it. The code editor has tabs for 'main.c' and 'main.o'. The code in 'main.c' is:

```
1 #include <stdio.h>
2
3 int main(){
4     int arr[5] = { -90, 78, 0, 56, 235 };
5     int peak = arr[0];
6     int pos = 0;
7
8     for (int i = 1; i < 5; i++) {
9         if (arr[i] > peak) {
10            peak = arr[i];
11            pos = i;
12        }
13    }
14
15    printf("Peak element of array is %d\n", peak);
16    printf("Position of peak is %d\n", pos);
17 }
```

The output window below the code editor displays the following text:

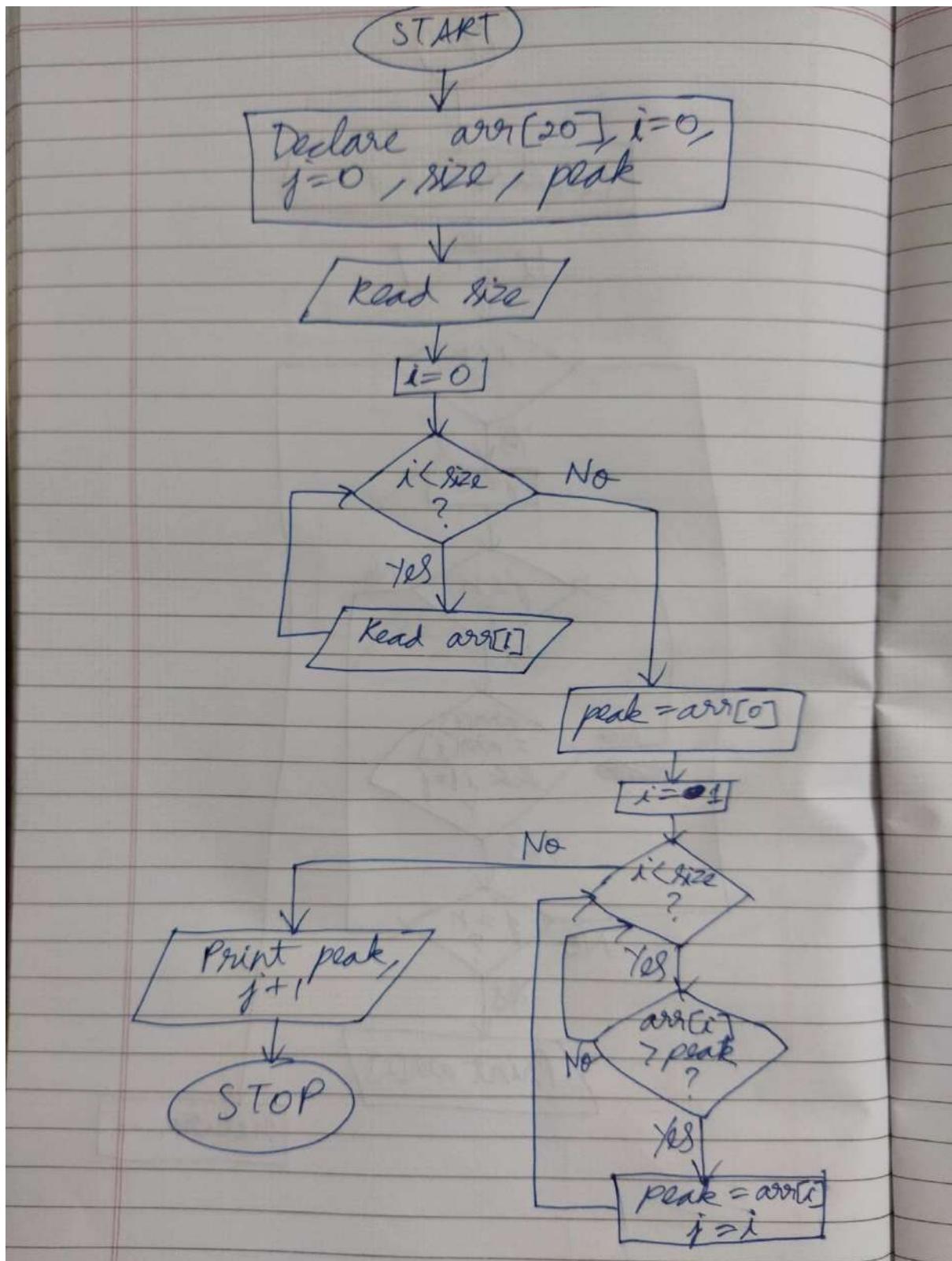
```
Enter array size : 5
Enter arr[0]: -90
Enter arr[1]: 78
Enter arr[2]: 0
Enter arr[3]: 56
Enter arr[4]: 235

Peak element of array is 235
Position of peak is 5

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- We are reading each array element while peak is initialized to arr[0], and if any array element is greater than current value of peak, value of peak will be set to that element and it's position is noted. Finally after reading all elements, peak element along with position is printed.

3) 1. START
2. Read size
3. for $i \leq 0$ to size do
4. read arr[i]
5. end for $i \leq size$
6. for $i \leq 1$ to size
7. If $arr[i] > peak$
8. peak $\leftarrow arr[i]$
9. $j \leftarrow i$
10. end for $i \leq size$
11. Print peak, $j+1$
12. STOP



4) Find the Kth largest and Kth smallest number in an array

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

```
int main(){
```

```
    int i,j,t,n,k,arr[20];
```

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

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

```
    printf("Enter array elements: \n");
```

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

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

```
    printf("Enter value of k: \n");
```

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

```
    if(k>n)
```

```
{
```

```
    printf("\n k value should not be greater than %d",n);
```

```
    return -1;
```

```
}
```

```
//sorting array in ascending order  
for (i=0;i<n;i++){  
    for(j=0;j<n-i-1;j++){  
        if (arr[j]>arr[j+1]){  
            t=arr[j];  
            arr[j]=arr[j+1];  
            arr[j+1]=t;  
        }  
    }  
}  
  
//printing sorted array  
printf("The sorted list is: ");  
for (i=0;i<n;i++){  
    printf("%d ", arr[i]);  
}  
  
printf("\n\nThe %dth largest element is %d \n",k, arr[n-k]);  
printf("The %dth smallest element is %d \n",k, arr[k-1]);  
  
return 0;
```

}

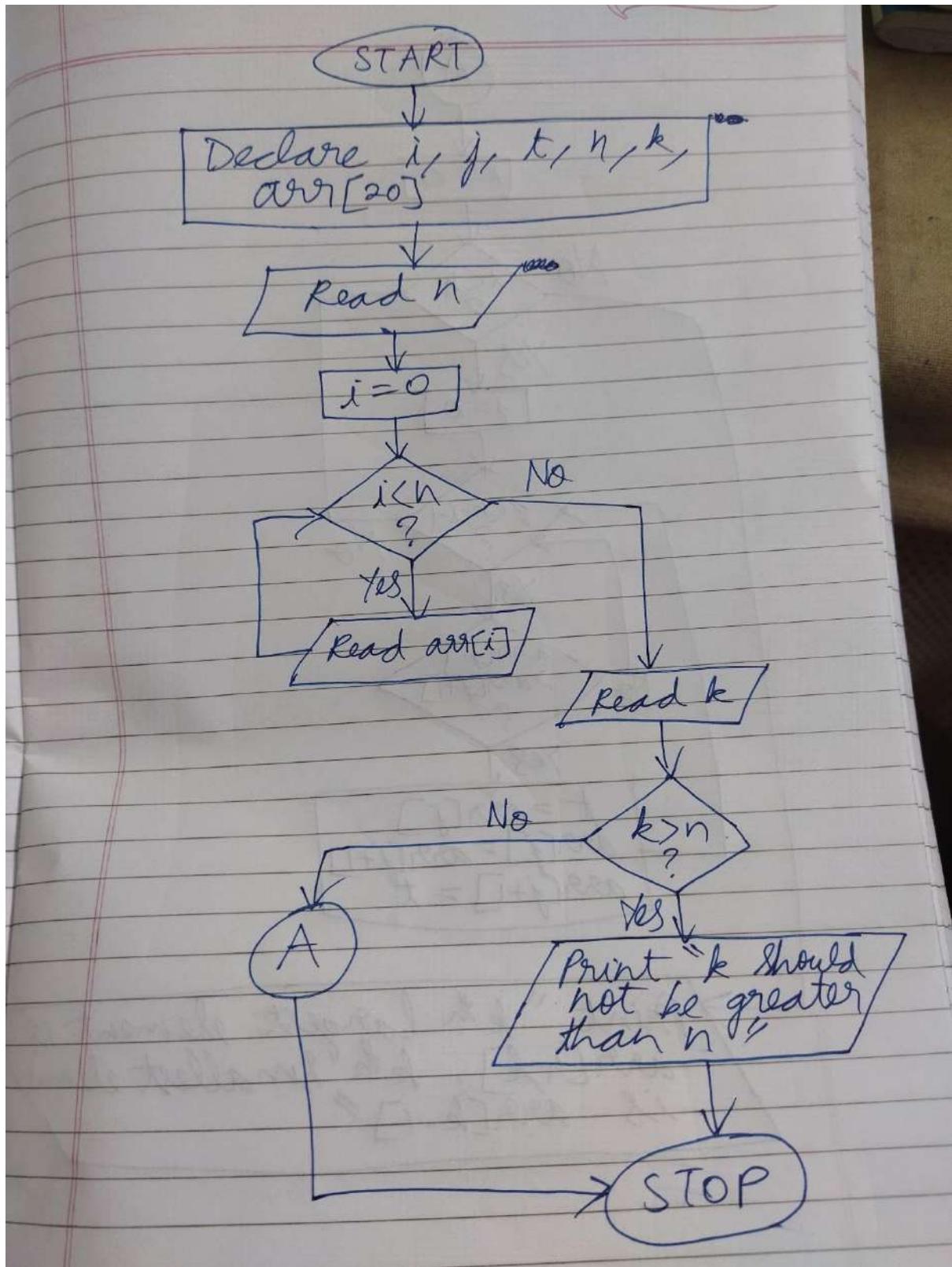
```
main.c
26     t=arr[j];
27     arr[j]=arr[j+1];
28     arr[j+1]=t;
29 }

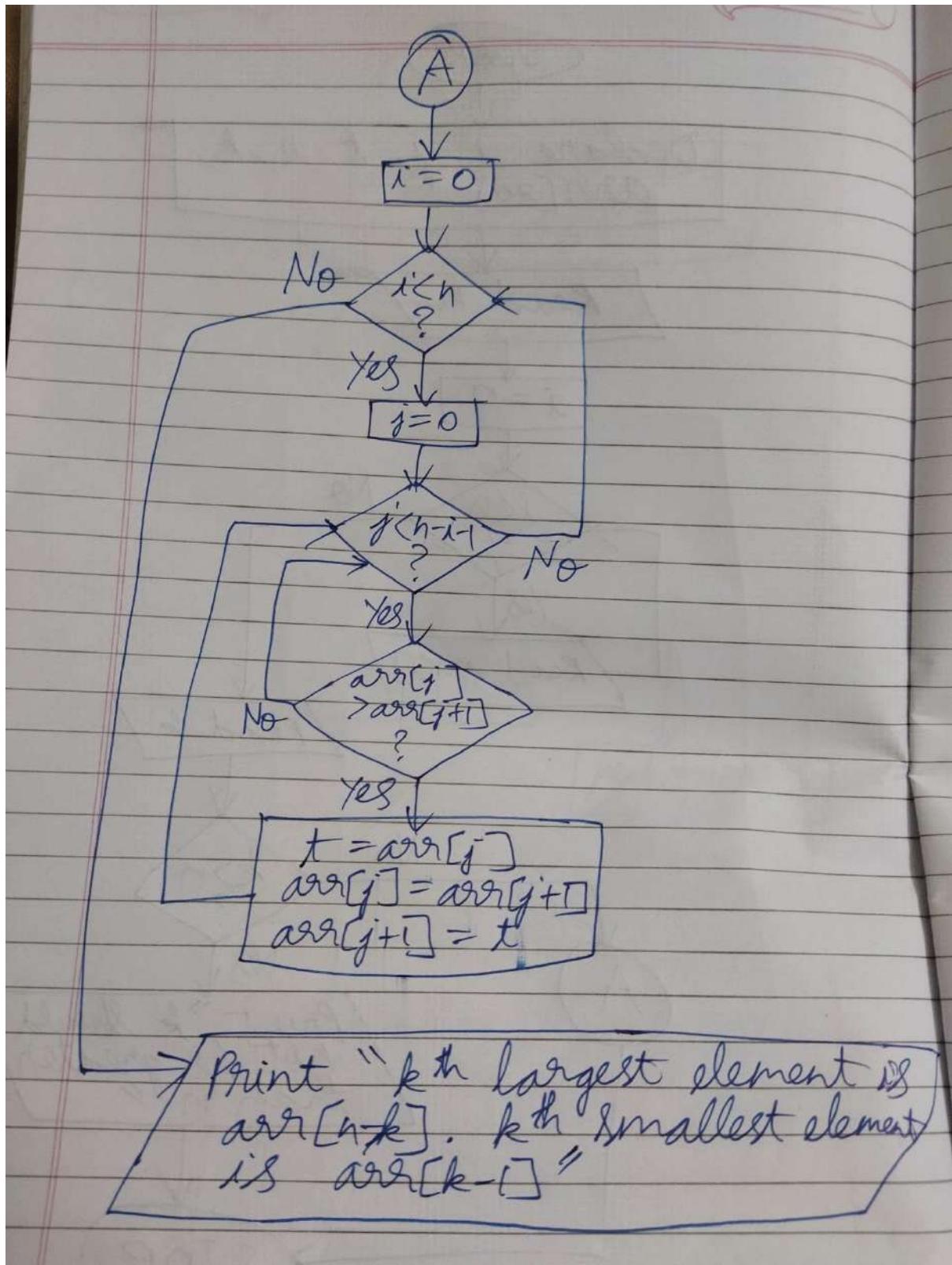
Enter size of array:
6
Enter array elements:
56
-89
90
34
0
8
Enter value of k:
4
The sorted list is: -89 0 8 34 56 90
The 4th largest element is 8
The 4th smallest element is 34

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- We first sort the array in ascending order, so kth largest element is the kth element from the end of sorted list, and kth smallest element is the kth element from the start of the sorted array.

4) 1. START
2. ~~Read~~ Read n
3. for $i \leftarrow 0$ to n do
4. Read arr[i]
5. end for $i \leftarrow n$
6. Read k
7. If $k > n$
8. Exit
9. for $i \leftarrow 0$ to n do
10. for $j \leftarrow 0$ to $n-i-1$ do
11. If arr[j] > arr[j+1]
12. t \leftarrow arr[j]
13. arr[j] \leftarrow arr[j+1]
14. arr[j+1] \leftarrow t
15. end for $j \leftarrow n-i-1$
16. end for $i \leftarrow n$
17. for $i \leftarrow 0$ to n do
18. Print arr[i]
19. end for $i \leftarrow n$
20. Print arr[n-k], arr[k-1]
21. STOP





5) Find the position of the pattern in a given string.

Ex: ABABABBBABCBBBBABCAAAAAAABC

Input: ABC

Pattern found at position 8, 16, 24

```
#include <stdio.h>
#include <string.h>
int main()
{
char text[20], pat[20];
int a,b;
printf("Enter the string :");
scanf("%s",text);
//printf("\nthe string is: %s",text);
printf("\nEnter the pattern to find :");
scanf("%s",pat);
//printf("\nthe pattern to find is: %s \n", pat);
a = strlen(pat);
b = strlen(text);
for (int i = 0; i <= b - a; i++) {
int j;
for (j = 0; j < a; j++)
if (text[i + j] != pat[j])
```

```

break;

if (j == a)
printf("Pattern found at position %d \n", i+1);

}

return 0;
}

```

```

main.c
1 #include <stdio.h>
2 #include <string.h>
3 int main()
4 {
5     Enter the string :abbcddbbfhigjbb
6
7     the string is: abbcddbbfhigjbb
8     Enter the pattern to find : bb
9
10    the pattern to find is: bb
11    Pattern found at position 2
12    Pattern found at position 7
13    Pattern found at position 14
14
15    ...Program finished with exit code 0
16    Press ENTER to exit console. []

```

Inference- Similar to finding pattern in previous lab assignment, but here we are printing all the positions where pattern is found.

5) 1. START

2. Read text, pat

3. a \leftarrow strlen(pat)

4. b \leftarrow strlen(text)

5. for i \leftarrow 0 to b - a do

6. for j \leftarrow 0 to a do

7. If $text[i+j] \neq pat[j]$

8. break

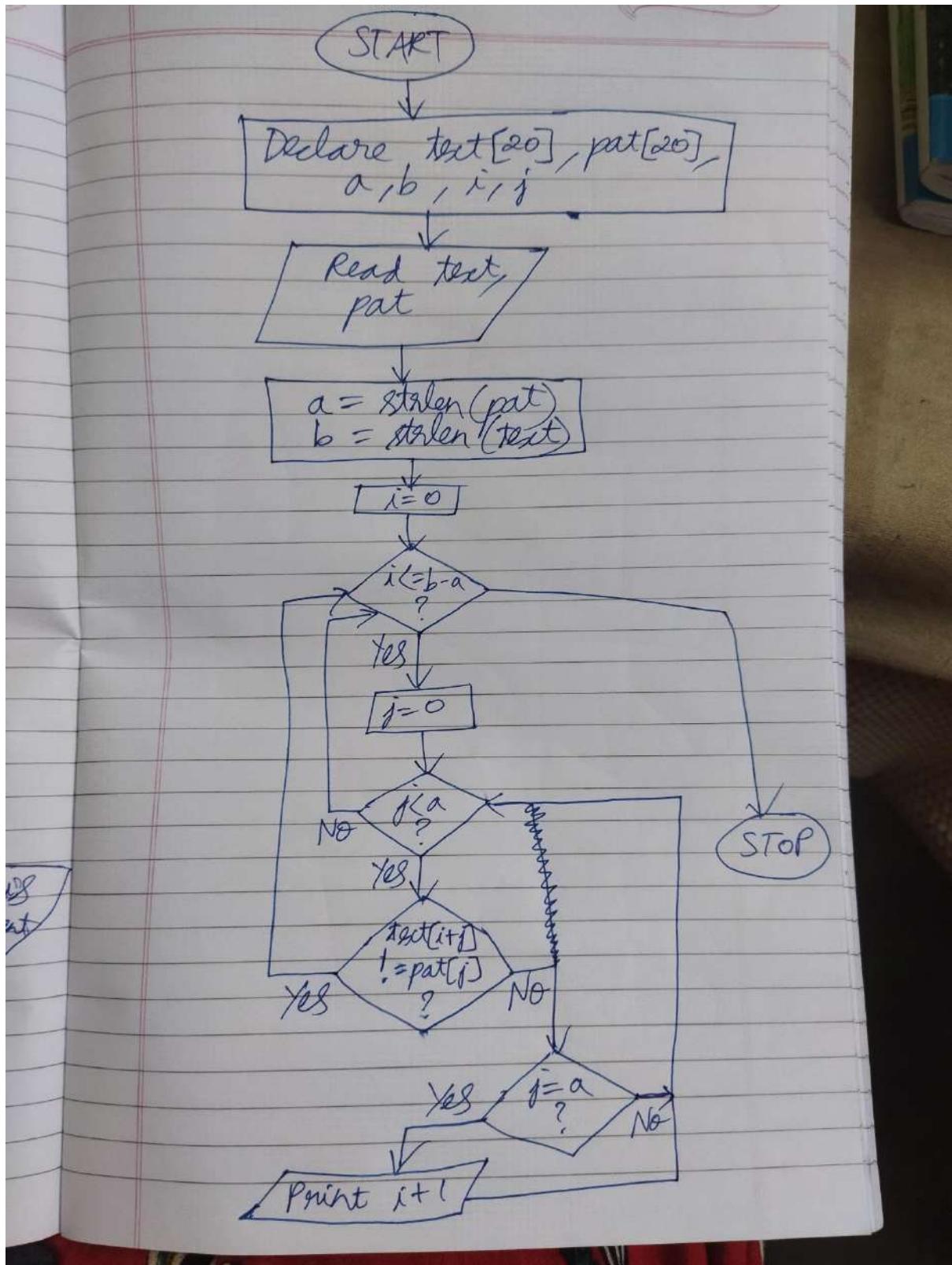
9. end for j \leftarrow a

10. If $i < a$

11. print i + 1

12. end for i $>$ b - a

13. STOP



6)Run Length Encoding,

If the input string is “wwwwaadexxxxx”, then the function
should return “w4a3d1e1x6”

```
#include <stdio.h>
#include <string.h>
int main()
{
char text[20];
int i, count, leng;
printf("Enter the string :");
scanf("%s",text);
//printf(" \nthe string is: %s\n",text);
```

```
leng = strlen(text)-1;
```

```
for ( i = 0; i <= leng; i++)
{
count =1;
while ( i <= leng && text[i] == text[i+1])
{
count++;
i++;
```

```

if (text[i] != text[i+1])
{
break;
}

}

printf("%c%d", text[i],count);
}

return 0;
}

```

```

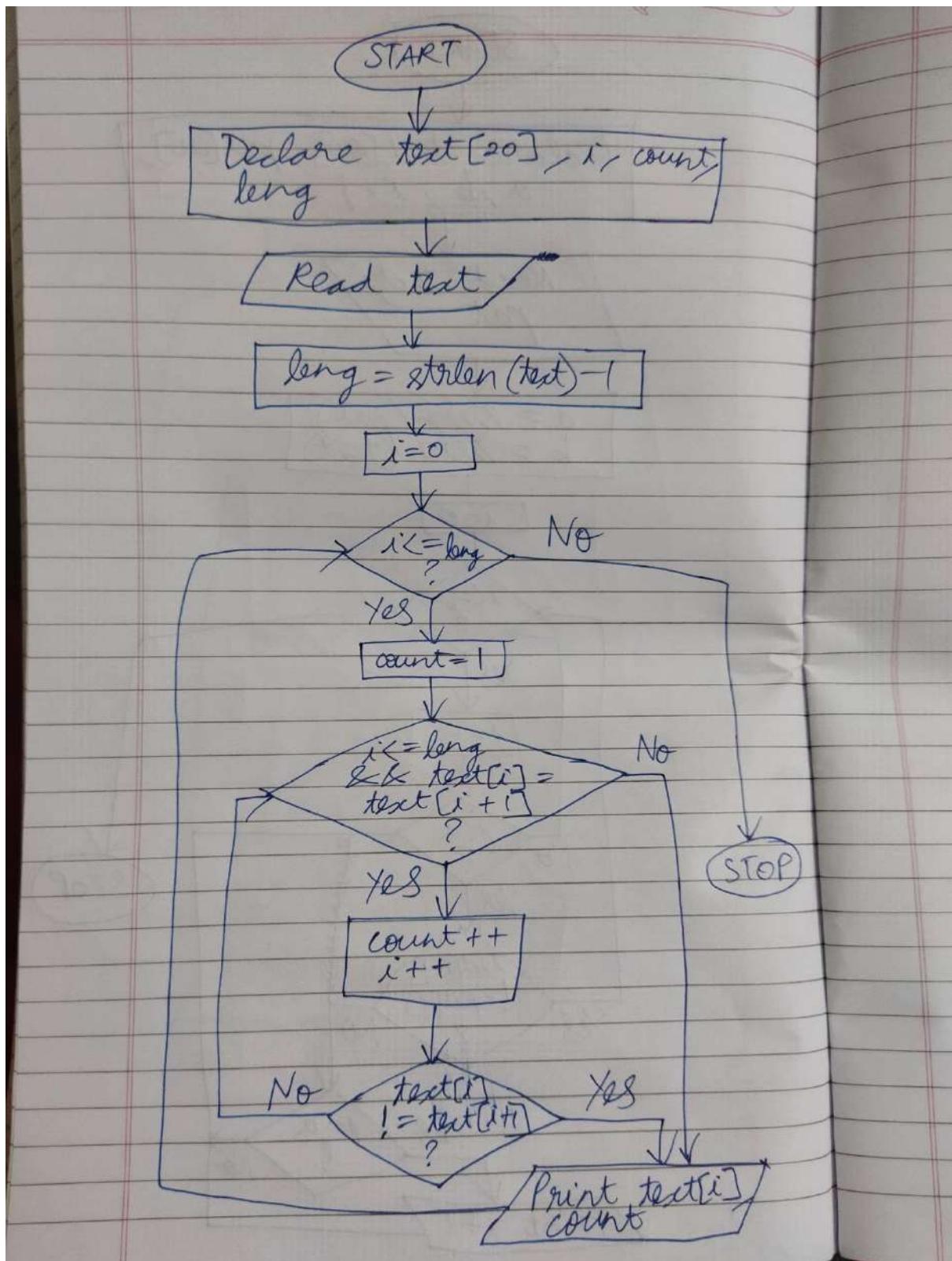
main.c
11 leng = strlen(text)-1;
12
13 for ( i = 0; i <= leng; i++)
Enter the string :wwwaaaadeeeeeeeexx
the string is: wwwaaaadeeeeeeeexx
w3a4d1e7x2
...Program finished with exit code 0
Press ENTER to exit console.

```

Inference- While the consecutive characters are the same, it'll keep counting the number of times it is repeated and

when the next consecutive character is different, it'll print the previous character along with the number of times it occurred consecutively, and this process is done for all characters in the string.

6) 1. START
2. Read text
3. leng < strlen(text) - 1
4. for i < 0 to leng do
5. count <= 1
6. while i <= leng AND text[i] = text[i+1]
7. count ++
8. i ++
9. If text[i] != text[i+1]
10. break
11. Print text[i], count
12. end for i > leng
13. STOP



LAB 9

Aim- Learn to use concept of strings and functions.

1. Given a string, find its first non-repeating character.

Input: arunkumar

Output: n

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

```
#include <string.h>
```

```
int main() {
```

```
    char str[100] = "applea";
```

```
    int len = strlen(str);
```

```
    int flag;
```

```
    /* Two loops to compare each  
character with other character  
*/
```

```
    for(int i = 0; i < len; i++) {
```

```
flag = 0;

for(int j = 0; j < len; j++) {

/* If it's equal and indexes
is not same */

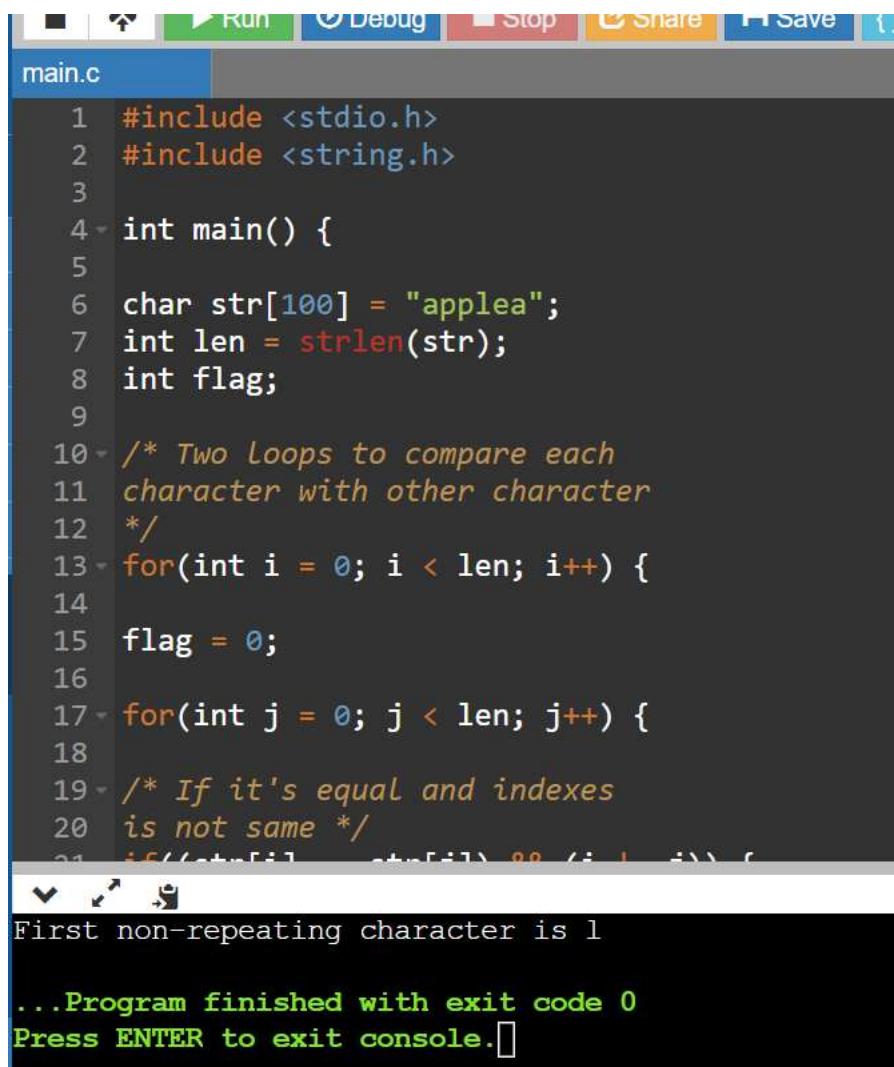
if((str[i] == str[j]) && (i != j)) {

flag = 1;
break;
}
}

if (flag == 0) {
printf("First non-repeating character is %c",str[i]);
break;
}

if (flag == 1) {
printf("Didn't find any non-repeating character");
}
```

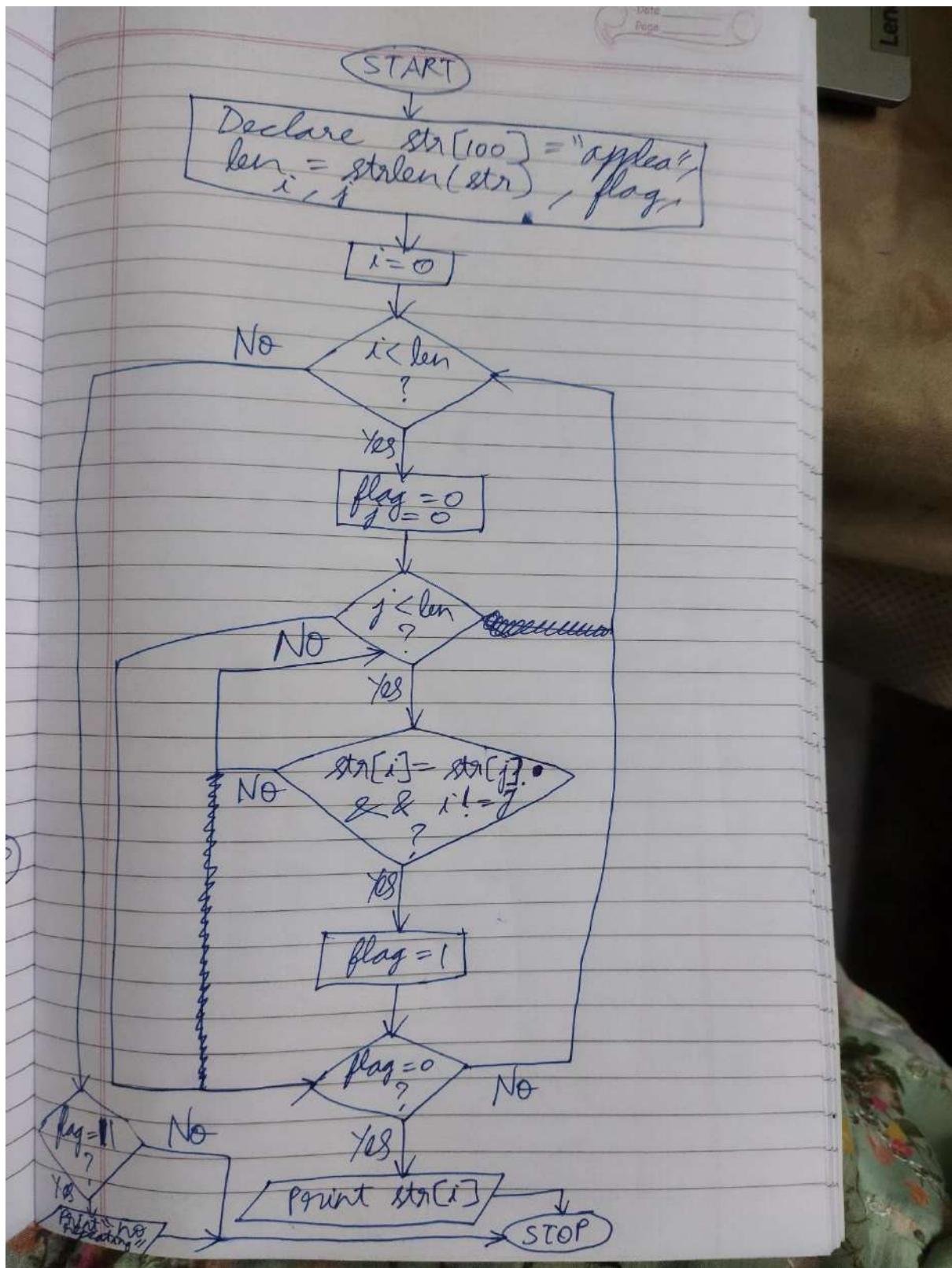
```
return 0;  
}  
  
}
```



```
Run Debug Stop Share Save ()  
main.c  
1 #include <stdio.h>  
2 #include <string.h>  
3  
4 int main() {  
5  
6 char str[100] = "applea";  
7 int len = strlen(str);  
8 int flag;  
9  
10 /* Two Loops to compare each  
11 character with other character  
12 */  
13 for(int i = 0; i < len; i++) {  
14  
15 flag = 0;  
16  
17 for(int j = 0; j < len; j++) {  
18  
19 /* If it's equal and indexes  
20 is not same */  
21 if(str[i] == str[j] && i != j) {  
22 flag = 1;  
23 }  
24 }  
25 if(flag == 0) {  
26 printf("First non-repeating character is %c\n", str[i]);  
27 }  
28 }  
29  
30 ...Program finished with exit code 0  
Press ENTER to exit console.[]
```

Interference- When it checks each character of string for repeating character and finds first repeating character, it won't check further and will print that character. If no repeating character is found, it'll so message.

1) 1. START
2. Read str
3. for $i \leftarrow 0$ to len do
4. flag $\leftarrow 0$
5. for $j \leftarrow 0$ to len do
6. If $str[i] \leftarrow str[j]$ AND $i \neq j$
7. break
8. end for $j \leftarrow len$
9. If flag $\leftarrow 0$
10. Print $str[i]$
11. break
12. end for $i \leftarrow len$
13. If flag $\leftarrow 1$
14. Print "No repeating characters"
15. STOP



2. Write a program to print all permutations of a given string.

Input: ABC

Output: ABC ACB BAC BCA CBA CAB

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

```
#include <string.h>
```

```
void changePosition(char *ch1, char *ch2)
```

```
{
```

```
char tmp;
```

```
tmp = *ch1;
```

```
*ch1 = *ch2;
```

```
*ch2 = tmp;
```

```
}
```

```
void charPermu(char *cht, int stno, int endno)
```

```
{
```

```
int i;
```

```
if (stno == endno)
```

```
printf("%s ", cht);
```

```
else
```

```
{
```

```
for (i = stno; i <= endno; i++)
```

```
{
```

```
changePosition((cht+stno), (cht+i));
```

```

charPermu(cth, stno+1, endno);
changePosition((cth+stno), (cth+i));
}
}

int main()
{
char str[] = "abcd";
int n = strlen(str);
printf(" The permutations of the string are : \n");
charPermu(str, 0, n-1);
printf("\n\n");
return 0;
}

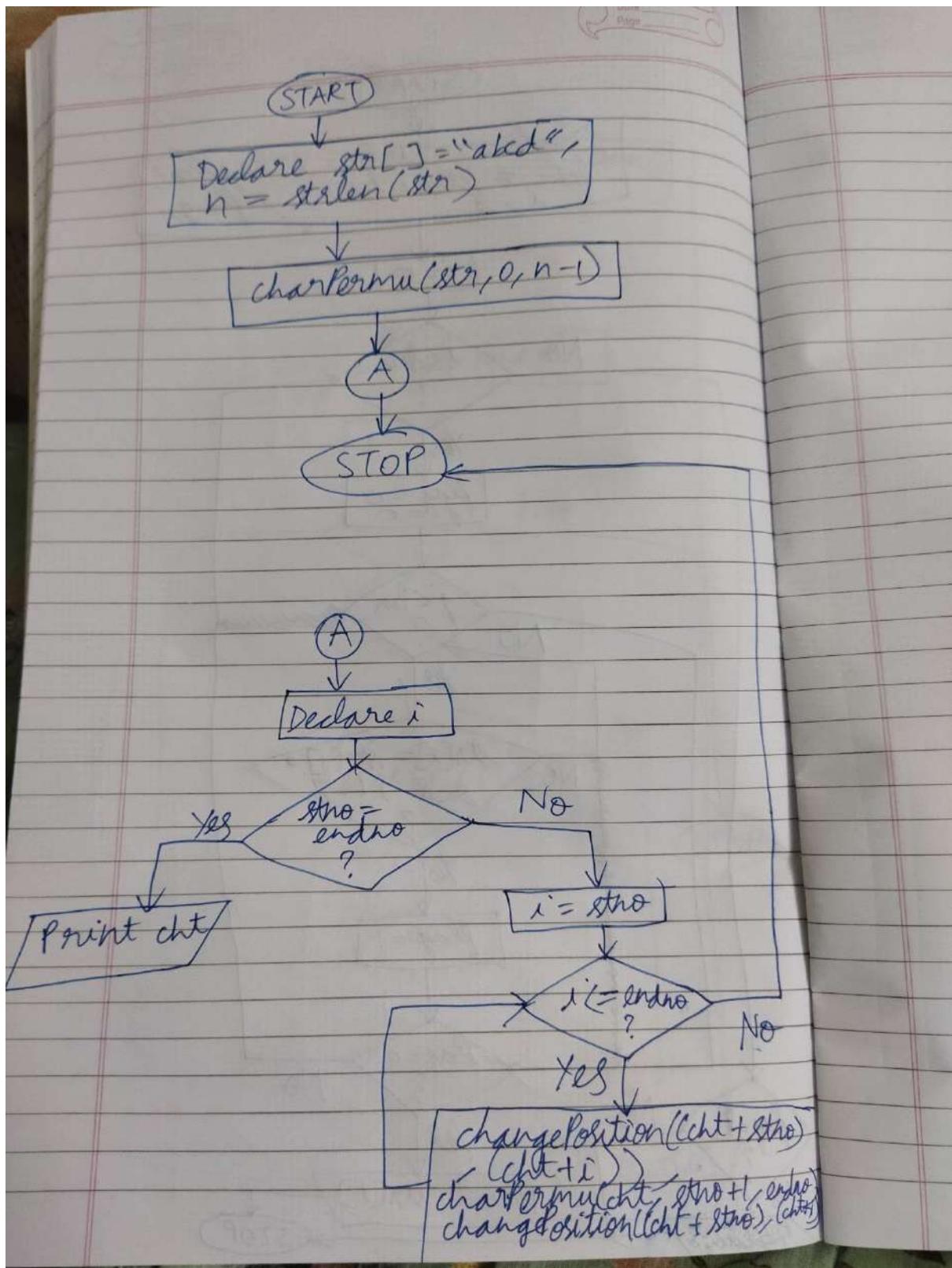
```

The screenshot shows a C IDE interface with the following details:

- Toolbar:** Run, Debug, Stop, Share, Save, Beautify.
- Language:** C.
- Code Editor:** The file "main.c" is open, showing lines 10 through 19 of the code.
- Output Window:** Labeled "input", displaying the output of the program. It shows the string "abcd" permuted in all possible ways, followed by "...Program finished with exit code 0 Press ENTER to exit console."

Inference- We are printing all permutations of the given string using recursion and pointers.

2) 1. START
2. Read str
3. Call charPermu(str, 0, n-1)
4. If stno < endno
5. Print cht
6. Else
7. for i < stno to endno do
8. call changePosition(cht+stno,
cht+i)
9. temp = *ch1
10. *ch1 = *ch2
11. *ch2 = temp
12. call charPermu(cht, stno+1, endno)
13. call changePosition(cht+stno,
cht+i)
14. end for i < endno + 1
15. STOP



3. Recursively removes all adjacent duplicates.

Input: abccb

Output: ay

First “abccb” is reduced to “abbd”.

The string “abbd” contains duplicates,
so it is further reduced to “ad”.

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

```
#include <string.h>
```

```
// Function to remove all adjacent duplicates from the given  
string
```

```
char* removeAdjDup(char* str, int n)  
{  
    // base case  
    if (n == 0) {  
        return str;  
    }
```

```
// `k` maintains the index of the next free location in the  
result,
```

```
//and `i` maintains the current index of the string
```

```
int i, k = 0;  
int len = strlen(str);
```

```
// start from the second character
for (i = 1; i < len; i++)
{
    // if the current character is not the same as the
    // previous character, add it to the result
    if (str[i - 1] != str[i]) {
        str[k++] = str[i - 1];
    }
    else {
        // remove adjacent duplicates
        while (i < len && str[i - 1] == str[i]) {
            i++;
        }
    }
}
```

```
// add the last character to the result
str[k++] = str[i - 1];
```

```
// null terminate the string
str[k] = '\0';
```

```
// start again if any duplicate is removed
if (k != n) {
    return removeAdjDup(str, k);
}

// if the algorithm didn't change the input string, that means
// all the adjacent duplicates are removed
return str;
}

int main(void)
{
    char str[] = "DBAABDAB";
    int n = strlen(str);

    printf("The string left after the removal of all adjacent
duplicates is
%s",
    removeAdjDup(str, n));

    return 0;
}
```

}

The screenshot shows a C programming environment with the following code in main.c:

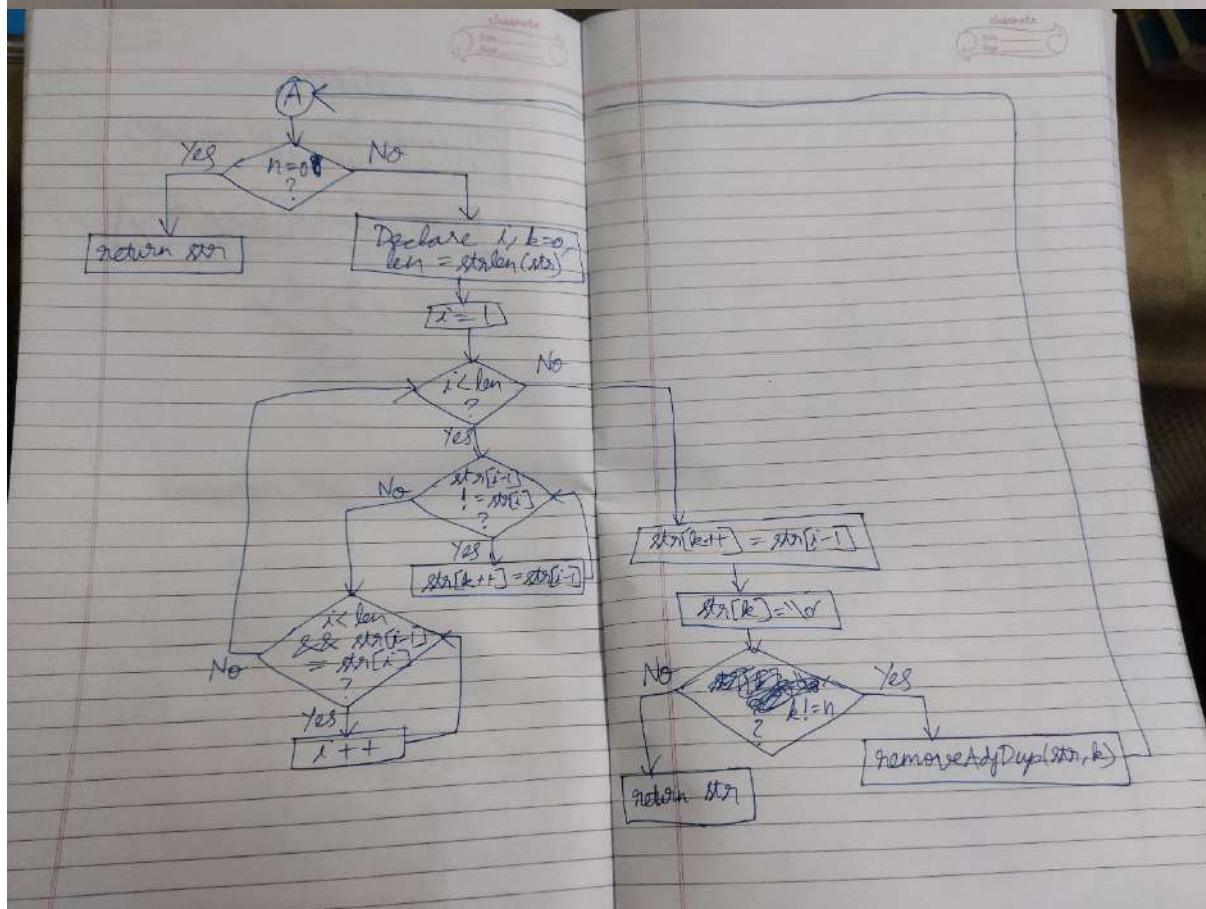
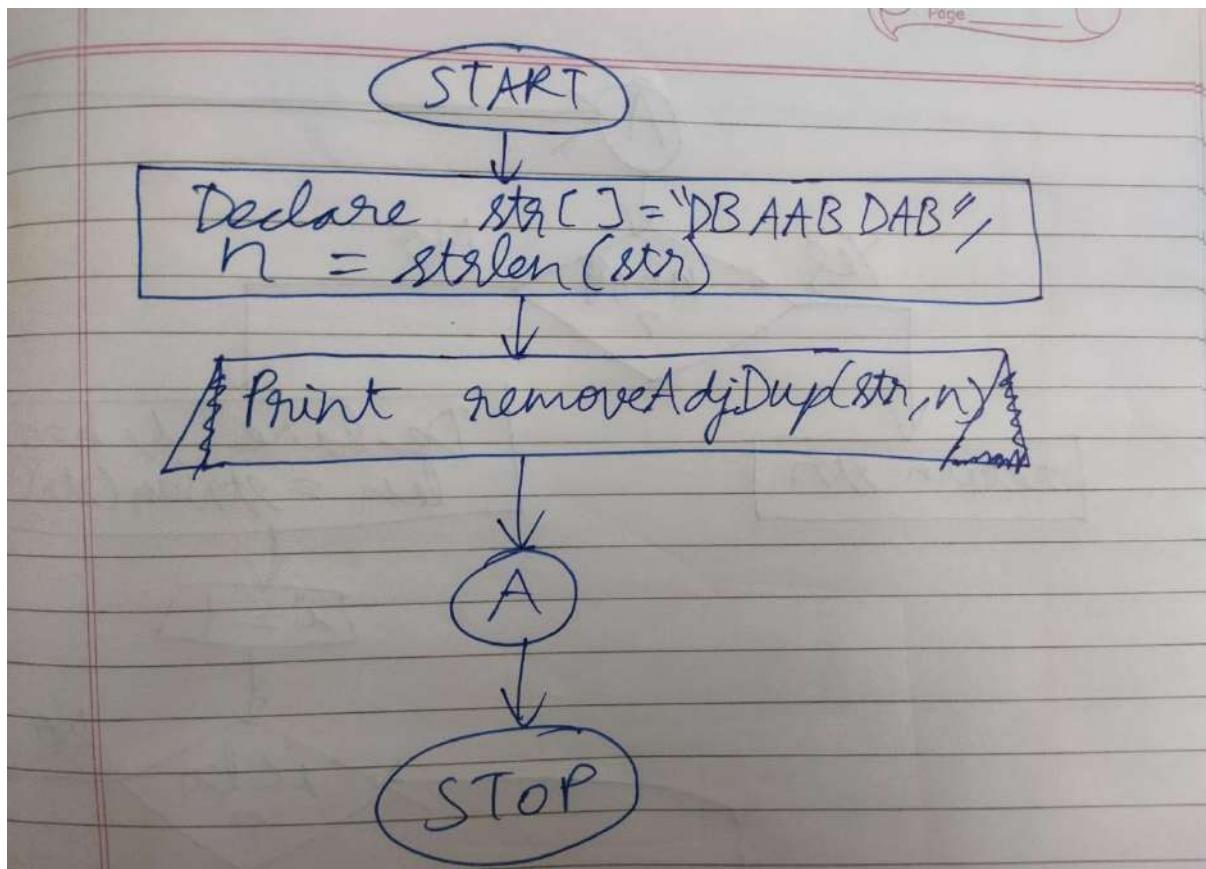
```
main.c
1 // If the algorithm didn't change the input string, that means
2 // all the adjacent duplicates are removed
3 return str;
4
5 int main(void)
6 {
7     char str[] = "DBAABDAB";
8     int n = strlen(str);
9
10    printf("The string left after the removal of all adjacent duplicates is %s",removeAdjDup(str, n));
11
12    return 0;
13 }
```

The output window shows the program's execution:

```
input
The string left after the removal of all adjacent duplicates is AB
...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- We are recursively removing all adjacent duplicate characters in string.

3) 1. START
2. $n \leftarrow \text{strlen}(\text{str})$
3. Print $\text{removeAdyDup}(\text{str}, n)$
4. If $n \leq 0$
5. return str
6. $\text{len} \leftarrow \text{strlen}(\text{str})$
7. for $i \leftarrow 1$ to len do
8. If $\text{str}[i-1] \neq \text{str}[i]$
9. $\text{str}[k+1] \leftarrow \text{str}[i-1]$
10. Else
11. while $i < \text{len}$ AND $\text{str}[i+1] \neq \text{str}[i]$
12. $i++$
13. end for $i \leftarrow \text{len}$
14. $\text{str}[k+1] \leftarrow \text{str}[\text{len}]$
15. $\text{str}[k] \leftarrow '\backslash 0'$
16. If $k = n$
17. return $\text{removeAdyDup}(\text{str}, k)$
18. return str
19. STOP



LAB 10

Aim- Learn to use concept of functions.

1. Fibonacci / Factorial: Develop functions to compute factorial n,
and the Fibonacci series till n terms using iteration.

Fibonacci

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

```
int fibonacci(int n){  
    int fib[25], i;  
  
    if(n==1)  
        printf("The Fibonacci series is: 0");  
  
    else if(n==2)  
        printf("The Fibonacci series is: 0, 1");  
  
    else{  
        fib[0]=0;  
        fib[1]=1;  
        printf("%d, %d", fib[0], fib[1]);  
        for(i=2; i<n; i++){
```

```
fib[i]=fib[i-1] + fib[i-2];
printf(", %d",fib[i]);
}
}
return fib[i];
}

int main()
{
int N=0;

printf("Enter number of terms in Fibonacci series: ");
scanf("%d", &N);

printf("The Fibonacci series is: ");
fibonacci(N);

return 0;
}
```

Factorial

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

```
int fact(int n){  
    int i=1;  
    long int f=1;  
  
    if(n==0)  
        printf("%ld",f);  
  
    else{  
        for(i=1;i<=n;i++){  
            f=f*i;  
        }  
        printf("%ld\n",f);  
    }  
  
}  
  
int main()  
{  
    int num;  
  
    printf("Enter number: ");  
    scanf("%d", &num);  
  
    if(num<0)
```

Samuela Abigail Mathew
71762108039

```

printf("Factorial of negative number does not exist");

else

printf("Factorial of %d is ",num);

fact(num);

return 0;

}

```

```

main.c
18 fib[i]=fib[i-1] + fib[i-2];
19 printf(", %d",fib[i]);
20 }
input
Enter number of terms in Fibonacci series: 12
The Fibonacci series is: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89
...Program finished with exit code 0
Press ENTER to exit console. []

```

```
main.c
15 }
16
17 }
18

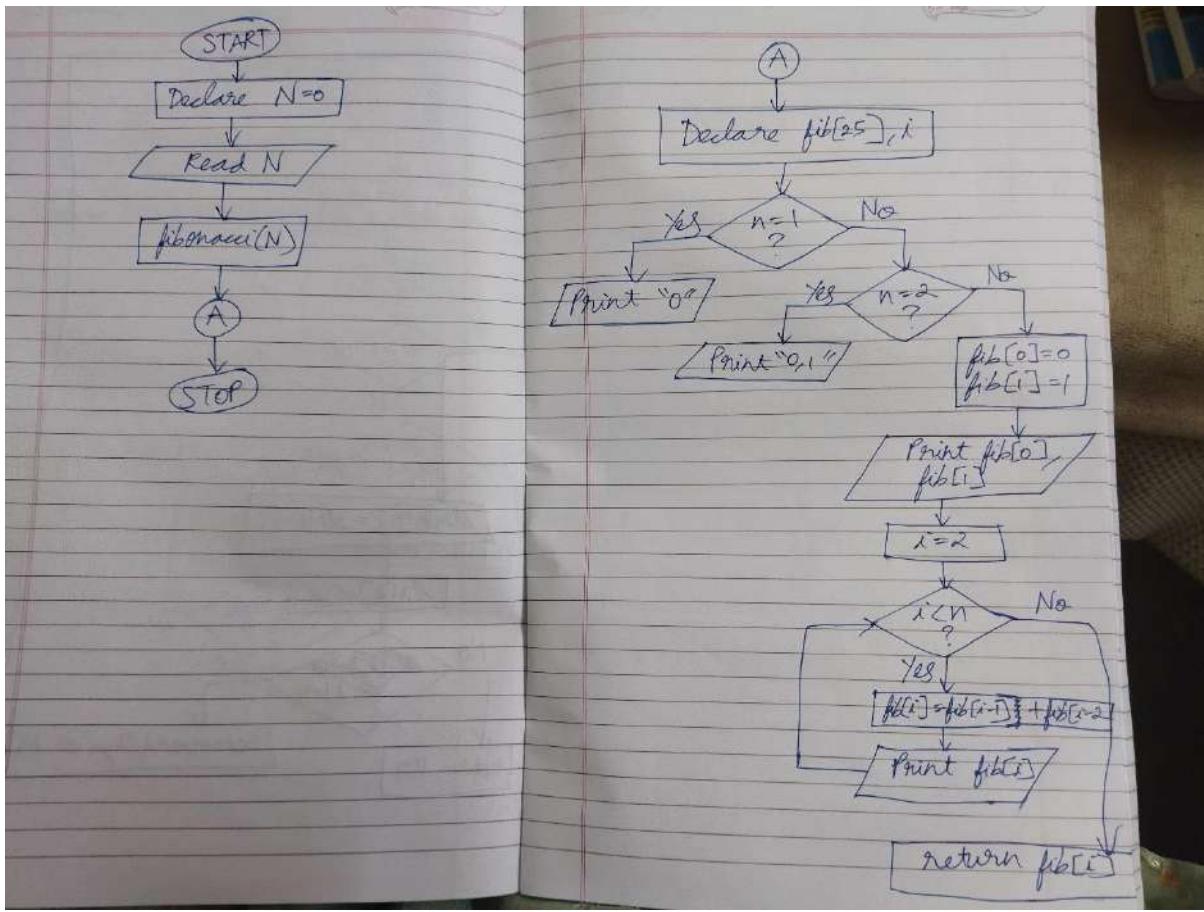
Enter number: 6
Factorial of 6 is 720

...Program finished with exit code 0
Press ENTER to exit console.
```

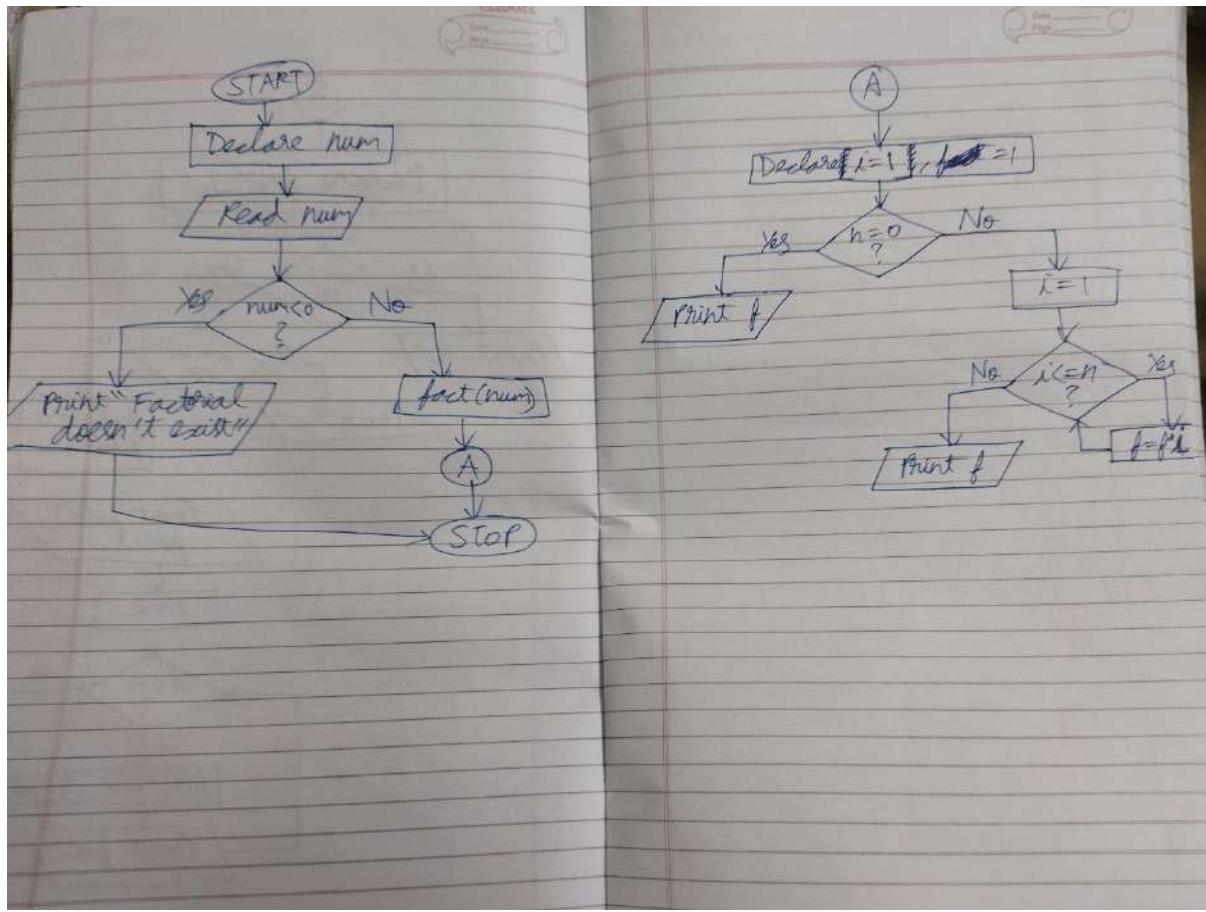
Inference- We have used functions to calculate Fibonacci series and factorial of a number.

Va)

1. START
2. Read N
3. call fibonaca(N)
4. If $n \leq 1$
5. Print 0
6. Else if $n \leq 2$
7. Print 0, 1
8. Else
9. $\text{fib}[0] \leftarrow 0$
10. $\text{fib}[1] \leftarrow 1$
11. Print $\text{fib}[0], \text{fib}[1]$
12. for $i \leftarrow 2$ to n do
13. $\text{fib}[i] \leftarrow \text{fib}[i-1] + \text{fib}[i-2]$
14. end for $i \leftarrow n$
15. return $\text{fib}[i]$ Print $\text{fib}[i]$
16. Print $\text{fib}[i]$
17. end for $i \leftarrow n$
18. return $\text{fib}[i]$
19. STOP



- b)
1. START
 2. Read num
 3. If num < 0
 4. Print "Factorial of negative numbers don't exist"
 5. Else
 6. call fact(num)
 7. If n < 0
 8. Print 1
 9. Else
 10. for i < 1 to n do
 11. f <= f * i
 12. end for i > n
 13. Print f
 14. STOP



2. Second Largest: Design a function that finds the second largest given 5 numbers into a function.

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

```
int find(int arr[5]) {
```

```
    int i, j, temp;
```

```
//sorting array in ascending order
```

```
    for (i=0;i<5;i++) {
```

```
        for(j=0;j<4-i;j++) {
```

```
            if (arr[j]>arr[j+1]) {
```

Samuela Abigail Mathew
71762108039

```
temp=arr[j];
arr[j]=arr[j+1];
arr[j+1]=temp;
}
}
}

printf("\nThe 2nd largest element is %d \n",arr[3]);
}
```

```
int main()
{
int a[5], i;

for(i=0; i<5;i++){
printf("Enter 5 array elements: ");
scanf("%d", &a[i]);
}
find(a);

return 0;
}
```

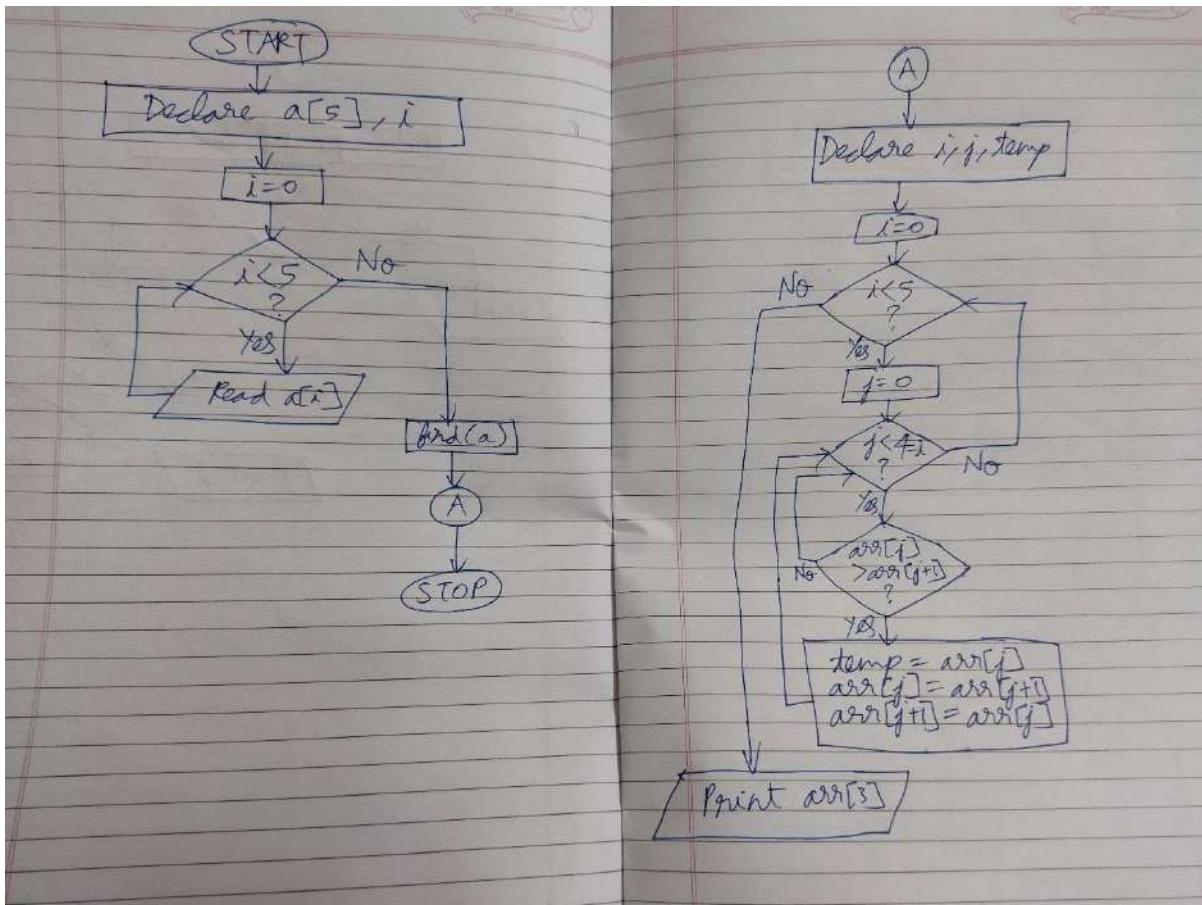
```
main.c
18 printf( "The sorted list is: " );
19 for (i=0;i<5;i++){
20 printf("%d ", arr[i]);
21 }

Enter 5 array elements: 5
Enter 5 array elements: 0
Enter 5 array elements: 3
Enter 5 array elements: 9
Enter 5 array elements: 2
The sorted list is: 0 2 3 5 9
The 2nd largest element is 5

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- We are passing the entire array into the function to find 2nd largest number.

1. START
 2. Read size
 3. for $i \leftarrow 0$ to size do
 4. Read $a[i]$
 5. end for $i \leftarrow size$
 6. call find(size, a)
 7. for $i \leftarrow 0$ to n do
 8. for $j \leftarrow 0$ to $n-i-1$ do
 9. If $arr[j] > arr[j+1]$
 10. temp $\leftarrow arr[j]$
 11. arr[j] $\leftarrow arr[j+1]$
 12. arr[j+1] $\leftarrow temp$
 13. end for $j \leftarrow n-i-1$
 14. end for $i \leftarrow n$
 15. for $i \leftarrow 0$ to n do
 16. Print $arr[i]$
 17. end for $i \leftarrow n$
 18. Print $arr[n-2]$
 19. STOP



3. Passing an Array: Convert the previous function in such a way that it finds and returns the second largest among n numbers to the main(). The numbers are passed as an array to the function.

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

```
int find(int n, int arr[25]) {
```

```
    int i, j, temp;
```

```
    //printing array in ascending order
```

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

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

```
            if (arr[j]>arr[j+1]){
```

```
                temp=arr[j];
```

```
                arr[j]=arr[j+1];
```

```
                arr[j+1]=temp;
```

```
}
```

```
}
```

```
}
```

```
    printf("\nThe 2nd largest element is %d \n",arr[n-2]);
```

```
    return arr[n-2];
```

```
}
```

```
int main()
```

Samuela Abigail Mathew
71762108039

```
{  
int a[25], i, size;  
  
printf("Enter array size: ");  
scanf("%d", &size);  
  
for(i=0; i<size;i++){  
printf("Enter array elements: ");  
scanf("%d", &a[i]);  
}  
  
find(size,a);  
return 0;  
}
```

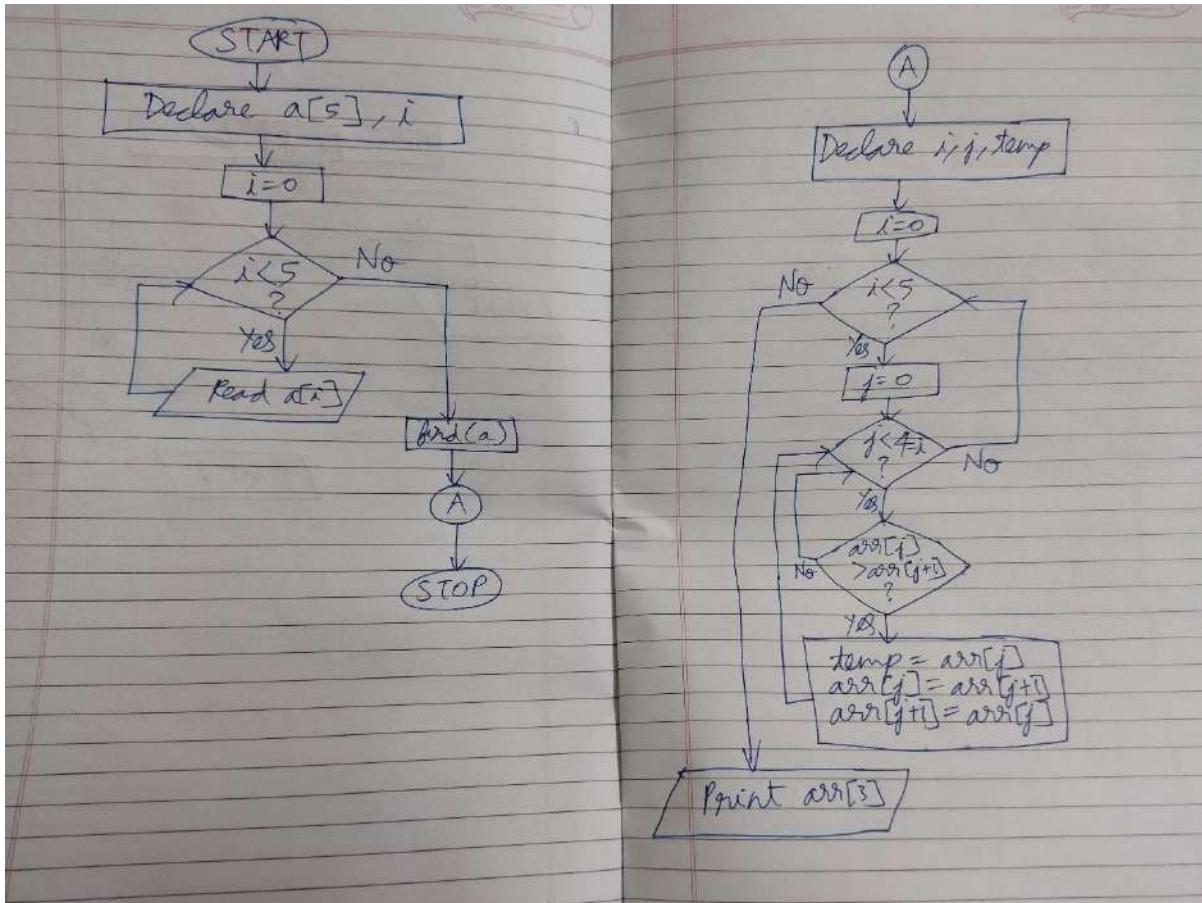
```
main.c
18 printf( "The sorted list is: " );
19 for (i=0;i<5;i++){
20 printf("%d ", arr[i]);
21 }

Enter 5 array elements: 5
Enter 5 array elements: 0
Enter 5 array elements: 3
Enter 5 array elements: 9
Enter 5 array elements: 2
The sorted list is: 0 2 3 5 9
The 2nd largest element is 5

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- We are passing the entire array into the function to find 2nd largest number.

1. START
 2. Read size
 3. for $i \leftarrow 0$ to size do
 4. Read $a[i]$
 5. end for $i \leftarrow size$
 6. call find(size, a)
 7. for $i \leftarrow 0$ to n do
 8. for $j \leftarrow 0$ to $n-i-1$ do
 9. If $arr[j] > arr[j+1]$
 10. temp $\leftarrow arr[j]$
 11. $arr[j] \leftarrow arr[j+1]$
 12. $arr[j+1] \leftarrow temp$
 13. end for $j \leftarrow n-i-1$
 14. end for $i \leftarrow n$
 15. for $i \leftarrow 0$ to n do
 16. Print $arr[i]$
 17. end for $i \leftarrow n$
 18. Print $arr[n-2]$
 19. STOP



4. Average: Write a function that computes the average of the numbers in an array of size 'n'. The array is passed as an argument to the function. No Global Variables!

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

```
int average(int arr[25], int size){
```

```
    int i;
```

```
    float j;
```

```
    if(size>0){
```

```
        for(i=1;i<size;i++){
```

```
            arr[i]+=arr[i-1];
```

```
            //printf("\n %d\n", arr[i]);
```

```
    }
```

```
//we are using size-1 instead of i as now i=size whereas
```

```
//in for loop we have condition i<size because a[i]=NUL which is 0
```

```
    j= arr[size-1]/size;
```

```
    printf("%f",j);
```

```
}
```

```
}
```

```
int main()
```

```
{
```

Samuela Abigail Mathew
71762108039

```
int a[25], siz=0, n;

printf("Enter array size: ");
scanf("%d", &siz);

for(n=0;n<siz;n++){
    printf("Enter array elements: ");
    scanf("%d", &a[n]);
}

printf("The average is: ");
average(a, siz);

return 0;
}
```

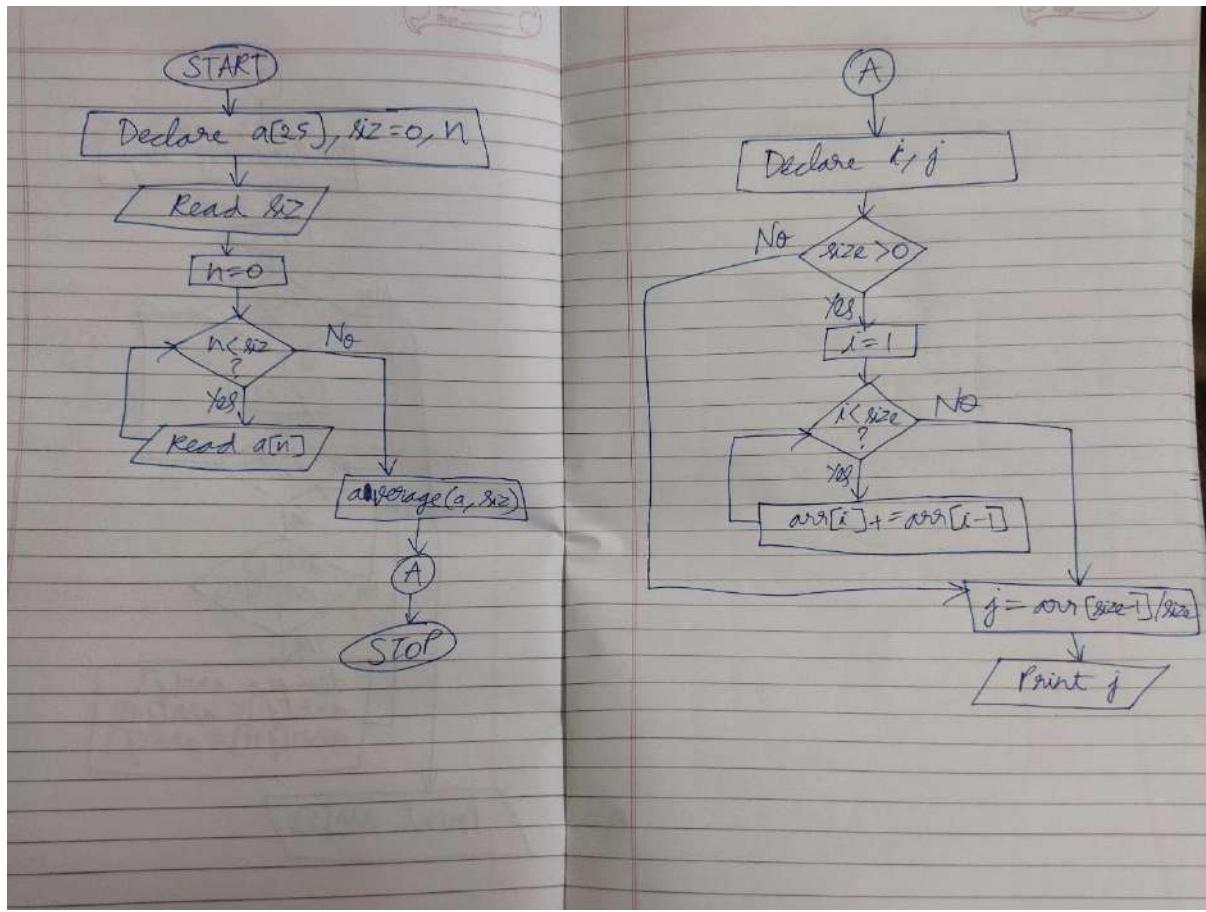
```
#include <stdio.h>
int average(int arr[25], int size){
    int i, sum = 0;
    for(i=0; i<size; i++)
        sum += arr[i];
    return sum / size;
}
int main() {
    int arr[25], size, i;
    printf("Enter array size: ");
    scanf("%d", &size);
    for(i=0; i<size; i++) {
        printf("Enter array elements: ");
        scanf("%d", &arr[i]);
    }
    printf("The average is: %.6f\n", average(arr, size));
    return 0;
}
```

Enter array size: 5
Enter array elements: 0
Enter array elements: 9
Enter array elements: -9
Enter array elements: 5
Enter array elements: 1
The average is: 1.000000

...Program finished with exit code 0
Press ENTER to exit console.

Inference- We are passing the entire array into function to find average.

4) 1. START
2. Read siz
3. for n <= 0 to siz do
4. Read a[n]
5. end for n <= siz
6. call average(a, siz)
7. If size > 0
8. for i <= 1 to size do
9. arr[i] += arr[i-1]
10. end for i <= size
11. j <- arr[size-1] / size
12. Print j
13. STOP



LAB 11

Aim- Learn to use concept of functions and recursion.

1. Complex Numbers: Write a function that takes 2 complex numbers as input and prints the sum and multiplication of the 2 complex numbers.

```
#include <stdio.h>
#include <stdlib.h>
struct complex
{
    int real, img;
};

void sum_mul(struct complex a, struct complex b){
    struct complex c;
    c.real = a.real + b.real;
    c.img = a.img + b.img;

    if (c.img >= 0){
        printf("\nSum of the complex numbers = %d + %di", c.real,
```

```
c.img);}

else{
printf("\nSum of the complex numbers = %d %di", c.real,
c.img);}

c.real = a.real*b.real + (-1*a.img*b.img);
c.img = a.img*b.real + a.real*b.img;

if (c.img >= 0){
printf("\nMultiplication of the complex numbers = %d + %di",
c.real, c.img);}
else{
printf("\nMultiplication of the complex numbers = %d %di",
c.real, c.img);}

}

int main()
{
struct complex a, b;
```

```
printf("Enter a and b where a + ib is the first complex  
number.");
```

```
printf("\na = ");  
scanf("%d", &a.real);
```

```
printf("b = ");  
scanf("%d", &a.img);
```

```
printf("Enter c and d where c + id is the second complex  
number.");
```

```
printf("\nc = ");  
scanf("%d", &b.real);
```

```
printf("d = ");  
scanf("%d", &b.img);
```

```
sum_mul(a, b);
```

```
return 0;
```

```
}
```

```
main.c
36 printf("Enter a and b where a + ib is the first complex number.\n");
37
38 scanf("%f", &a.real);
39 scanf("%f", &a.img);

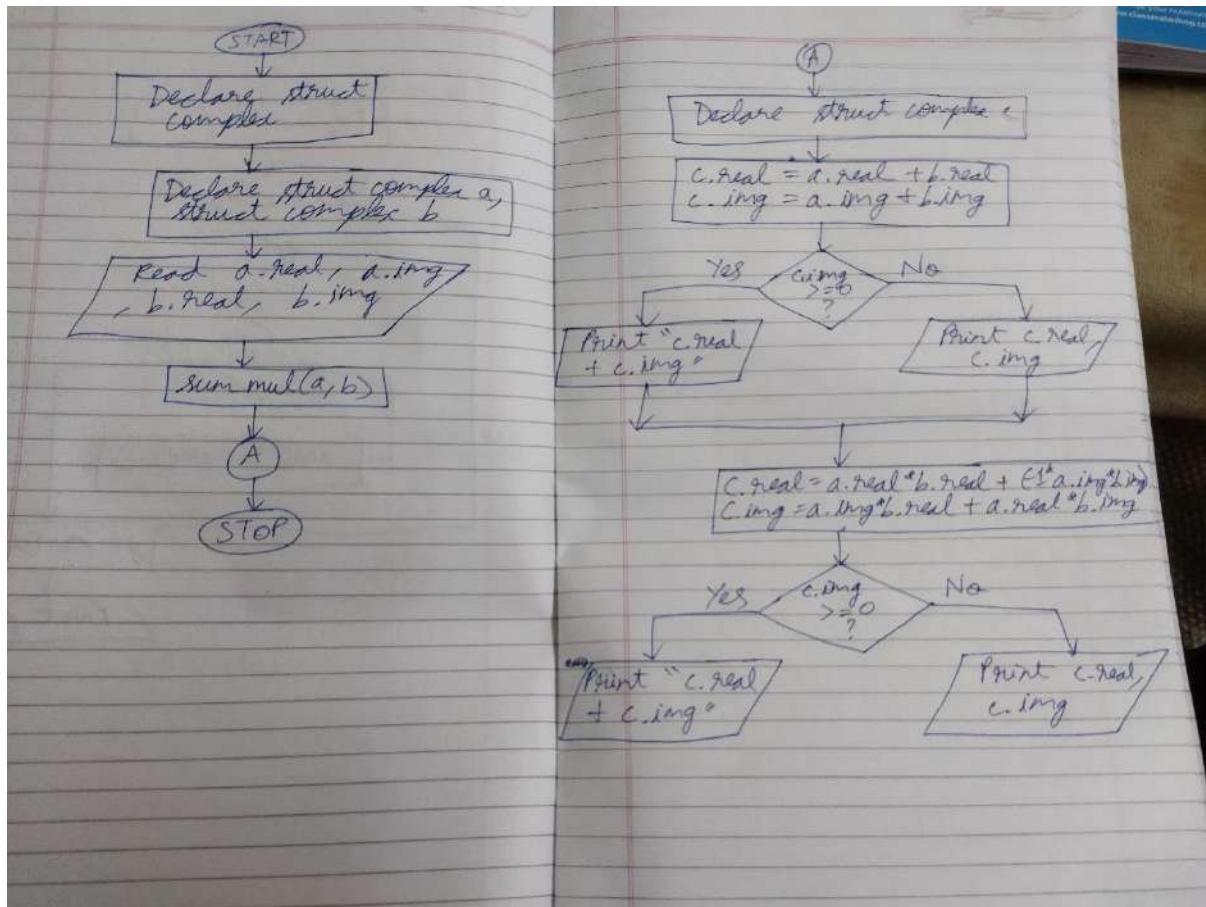
Enter a and b where a + ib is the first complex number.
a = 2
b = 4
Enter c and d where c + id is the second complex number.
c = -9
d = 23

Sum of the complex numbers = -7 + 27i
Multiplication of the complex numbers = -110 + 10i

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- We have used structures and function concept to calculate sum and product of 2 complex numbers.

1. START
2. Read a.real, a.img, b.real, b.img
3. call sum mul(a,b)
4. c.real \leftarrow a.real + b.real
5. c.img \leftarrow a.img + b.img
6. If c.img ≥ 0
7. Print c.real + c.img i
8. Else
9. Print c.real c.img i
10. c.real \leftarrow a.real * b.real + (-1 * a.img * b.img)
11. c.img \leftarrow a.img * b.real + a.real * b.img
12. If c.img ≥ 0
13. Print c.real + c.img i
14. Else
15. Print c.real c.img i
16. STOP



2. Fibonacci Analyzation: Compute Fibonacci(n) given n . How many calls are required for obtaining this n th number in the series? Draw a recurrence tree for the same.

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

```
int fibonacci(int i) {
    if(i == 0) {
        return 0;
    }
```

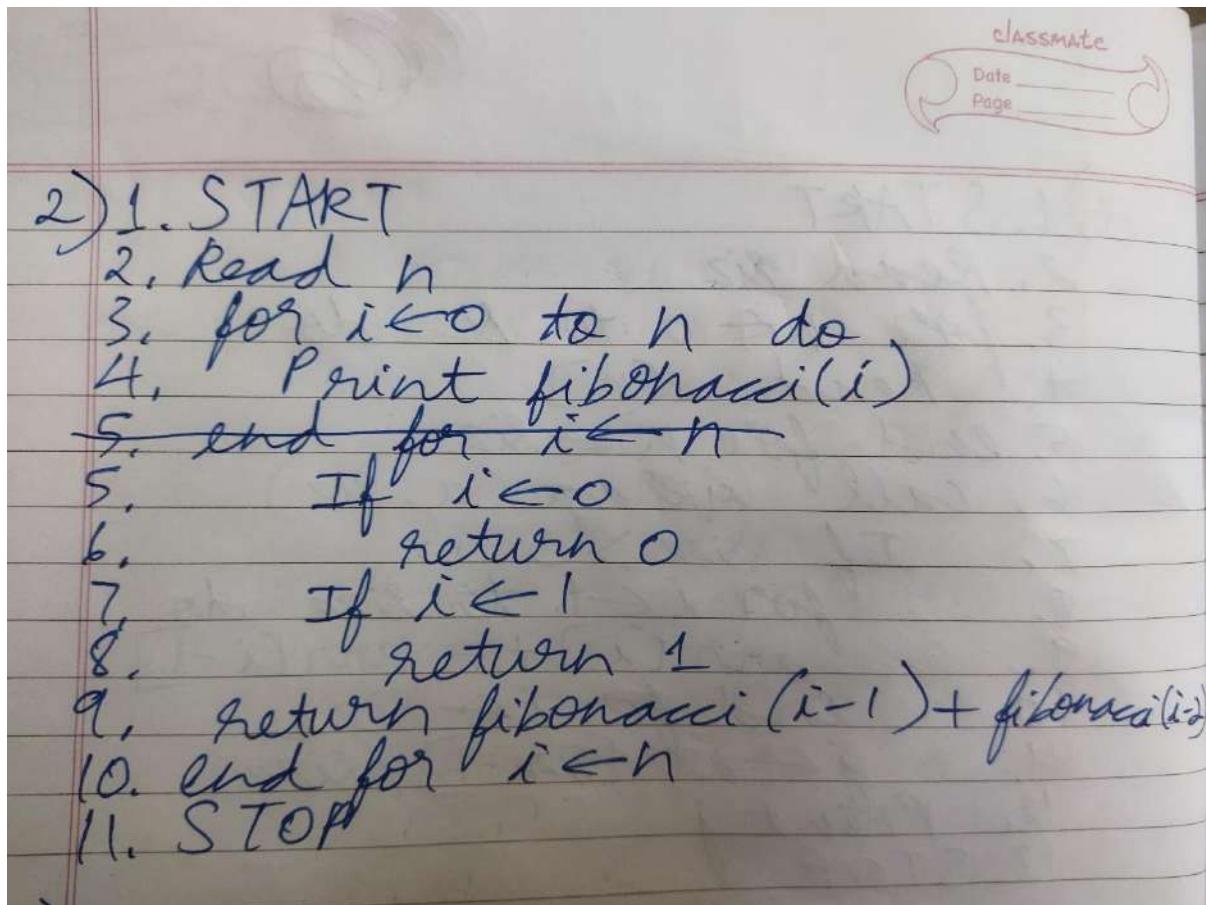
```
if(i == 1) {  
    return 1;  
}  
return fibonacci(i-1) + fibonacci(i-2);  
}  
  
int main() {  
  
    int i, n;  
  
    printf("Enter number of terms: ");  
    scanf("%d", &n);  
  
    printf("The Fibonacci series upto %d terms:\n",n);  
    for (i = 0; i < n; i++) {  
        printf("%d\n", fibonacci(i));  
    }  
    return 0;  
}
```

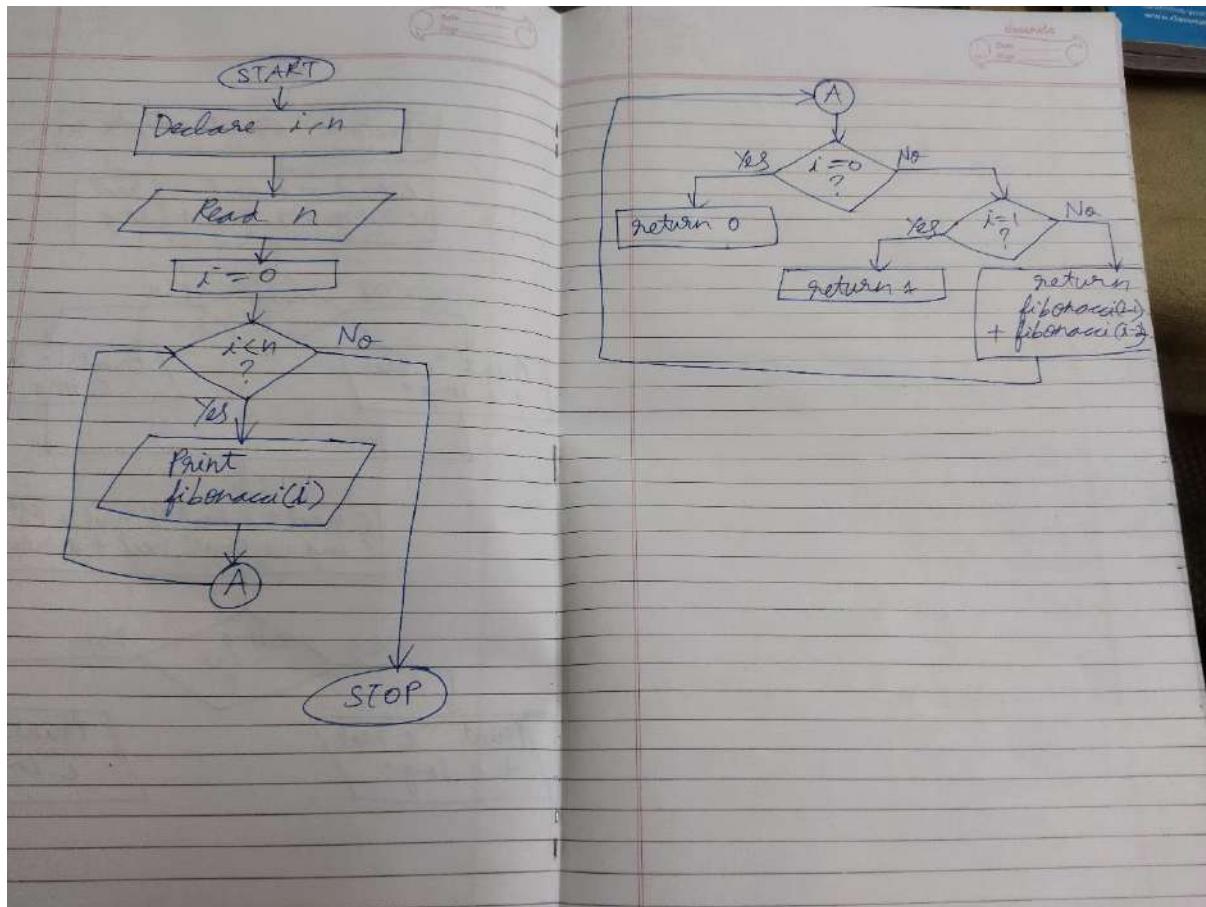
The screenshot shows a terminal window with the following content:

```
main.c
18 fib[i]=fib[i-1] + fib[i-2];
19 printf(", %d",fib[i]);
20 }

Enter number of terms in Fibonacci series: 12
The Fibonacci series is: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89
...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- We have used recursion to generate Fibonacci series.





3. 3-Way Merge: Given three sorted arrays of sizes m, n and o, write a function that merges the three into another array of size $m + n + o$ such that this new array also remains sorted.

```

#include<stdio.h>

void mergeArray(int a[], int n1, int b[], int n2, int c[], int n3,
int mer[], int n4)

{
    int i=0, j=0, k=0, l=0, m=0, temp;

    for(i=0;i<n1;i++) //Array Initialized
  
```

```
{  
mer[k]=a[i];  
k++;  
}  
  
l=n1;  
for(i=0;i<n2;i++) //Array Initialized  
{  
mer[l]=b[i];  
l++;  
}  
  
m=n1+n2;  
for( i=0;i<n3;i++) //Array Initialized  
{  
mer[m]=c[i];  
m++;  
}  
  
//mer[n4]=mer[m];  
printf("\nThe merged array is\n");  
for(int i=0;i<n4;i++){
```

```
printf("%d ",mer[i]);  
}
```

```
printf("\nAfter sorting the sorted array is\n");  
for(int i=0;i<n4;i++) //sorts in descending order  
{  
    for(int j=i+1; j<n4 ;j++)  
    {  
        if(mer[i]<mer[j])  
        {  
            temp=mer[i];  
            mer[i]=mer[j];  
            mer[j]=temp;  
        }  
    }  
}
```

```
for(int i=0 ; i<n4 ; i++)  
{  
    printf(" %d ",mer[i]);  
}  
}
```

Samuela Abigail Mathew
71762108039

```
int main()
{
    int n1, n2, n3, n4, i;

    printf("\nEnter size of First Array : ");
    scanf("%d", &n1);

    int a[n1];
    printf("\nEnter the elements for First Array : ");
    for(i = 0; i < n1; i++)
    {
        scanf("%d", &a[i]);
    }

    printf("\nEnter size of second Array : ");
    scanf("%d", &n2);

    int b[n2];
    printf("\nEnter the elements for Second Array : ");
    for(i = 0; i < n2; i++)
    {
```

```
    scanf("%d", &b[i]);  
}  
  
printf("\nEnter size of third Array : ");  
scanf("%d", &n3);
```

```
int c[n3];  
printf("\nEnter the elements for third Array : ");  
for(i = 0; i < n3; i++)  
{  
    scanf("%d", &c[i]);  
}
```

```
n4 = n1 + n2+ n3;  
int mer[n4];  
mergeArray(a, n1, b, n2, c, n3, mer, n4); //Function Call  
return 0;  
}
```

```
main.c
74 for(i=0, i < n2, i++)
75 {
76     scanf("%d", &b[i]);
77 }

3
Enter size of second Array : 5
Enter the elements for Second Array : 12
-90
67
3
0

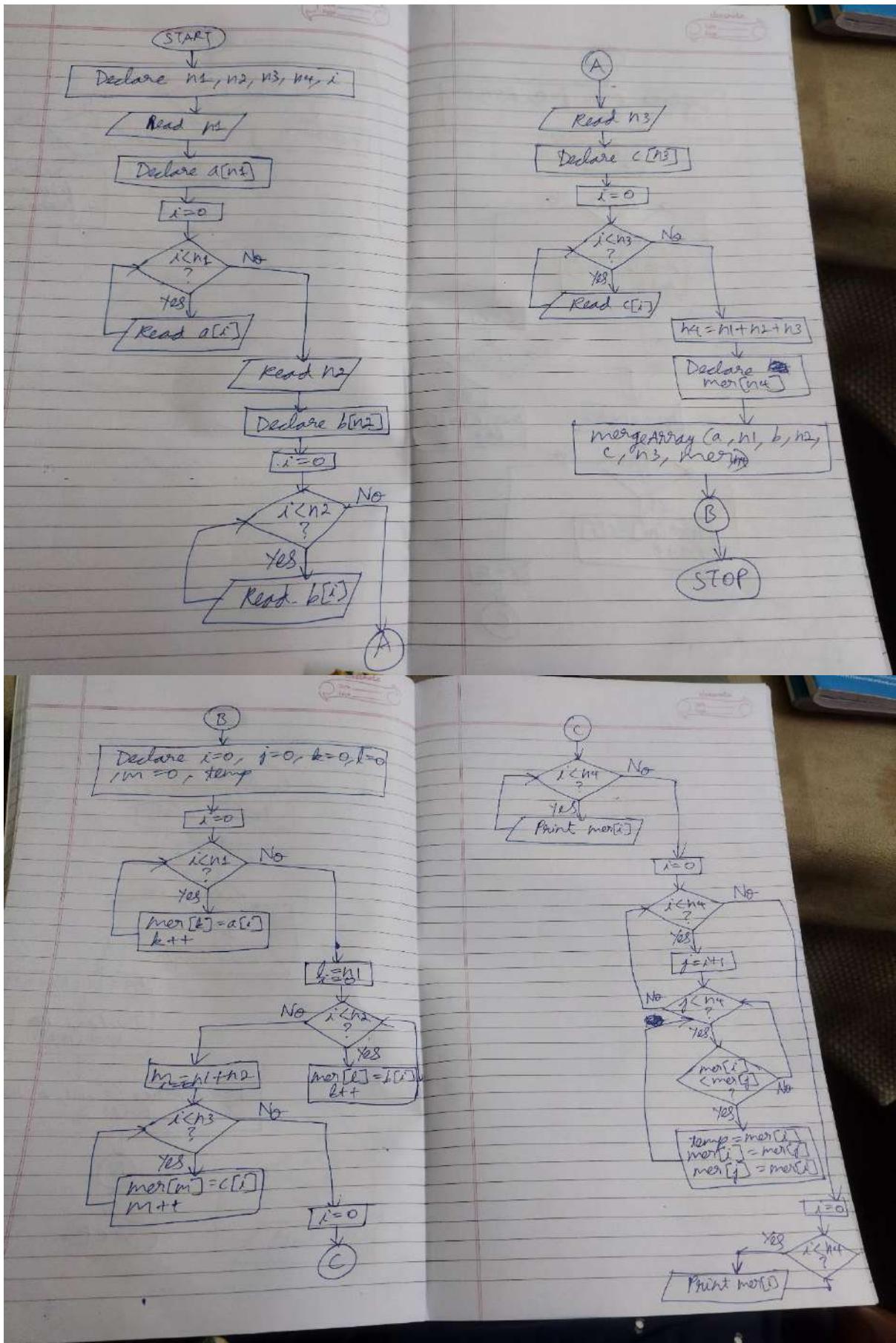
Enter size of third Array : 4
Enter the elements for third Array : 67
24
-789
274

The merged array is
34 90 3 12 -90 67 3 0 67 24 -789 274
After sorting the sorted array is
274 90 67 67 34 24 12 3 3 0 -90 -789
...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- We have used functions to merge three arrays into a single sorted array.

3) 1. START
 2. Read n_1, n_2, n_3
 3. for $i \leftarrow 0$ to n_1 do
 4. Read $a[i]$
 5. end for $i \leftarrow n_1$
 6. for $i \leftarrow 0$ to n_2 do
 7. Read $b[i]$
 8. end for $i \leftarrow n_2$
 9. for $i \leftarrow 0$ to n_3 do
 10. Read $c[i]$
 11. end for $i \leftarrow n_3$
 12. $n_4 \leftarrow n_1 + n_2 + n_3$
 13. call mergeArray ($a, n_1, b, n_2,$
 c, n_3, mer)
 14. for $i \leftarrow 0$ to n_1 do
 15. $\text{mer}[k] = a[i]$
 16. $k++$
 17. end for $i \leftarrow n_1$
 18. $l \leftarrow n_1$
 19. for $i \leftarrow 0$ to n_2 do
 20. $\text{mer}[l] = b[i]$

21. $l++$
 22. end for $i \leftarrow n_2$
 23. $m \leftarrow n_1 + n_2$
 24. for $i \leftarrow 0$ to n_3 do
 25. $\text{mer}[m] = c[i]$
 26. $m++$
 27. end for $i \leftarrow n_3$
 28. for $i \leftarrow 0$ to n_4 do
 29. Print $\text{mer}[i]$
 30. end for $i \leftarrow n_4$
 31. for $i \leftarrow 0$ to n_4 do
 32. for $j \leftarrow i+1$ to n_4 do
 33. If $\text{mer}[i] < \text{mer}[j]$
 34. temp $\leftarrow \text{mer}[i]$
 35. $\text{mer}[i] \leftarrow \text{mer}[j]$
 36. $\text{mer}[j] \leftarrow \text{temp}$
 37. end for $j \leftarrow n_4$
 38. end for $i \leftarrow n_4$
 39. for $i \leftarrow 0$ to n_4 do
 40. Print $\text{mer}[i]$
 41. end for $i \leftarrow n_4$
 42. STOP



4. Rearrangement: Given an array A[1:n] which contains a set of two characters ('B' and 'G') representing Boys and Girls standing in a row in no particular order. Devise a function that rearranges boys after all the girls in the row.
Remember, your code should perform single-scan on that entire array!

```
#include <stdio.h>
#include <ctype.h>
int main()
{
char name[] = {'B','G','G','B','G','G'};
int size=0, i=0, count=0;
size = sizeof(name);
while(name[i] != '\0')
{
    i++;
    if(name[i]=='G')
        ++count;
}
printf("count of girls = %d, size=%d \n", count, size);
for (i=0; i<count; i++)
    name[i]='G';
for (i=count; i<size; i++)
```

```

name[i]='B';

i=0;

while(i<size)
{
    printf("%c", name[i]);

    i++;

}

return 0;
}

```

The screenshot shows a terminal window with the following content:

```

main.c
8 while(name[i] != 'G')
9 {
10     i++;
11     if(name[i]=='G')

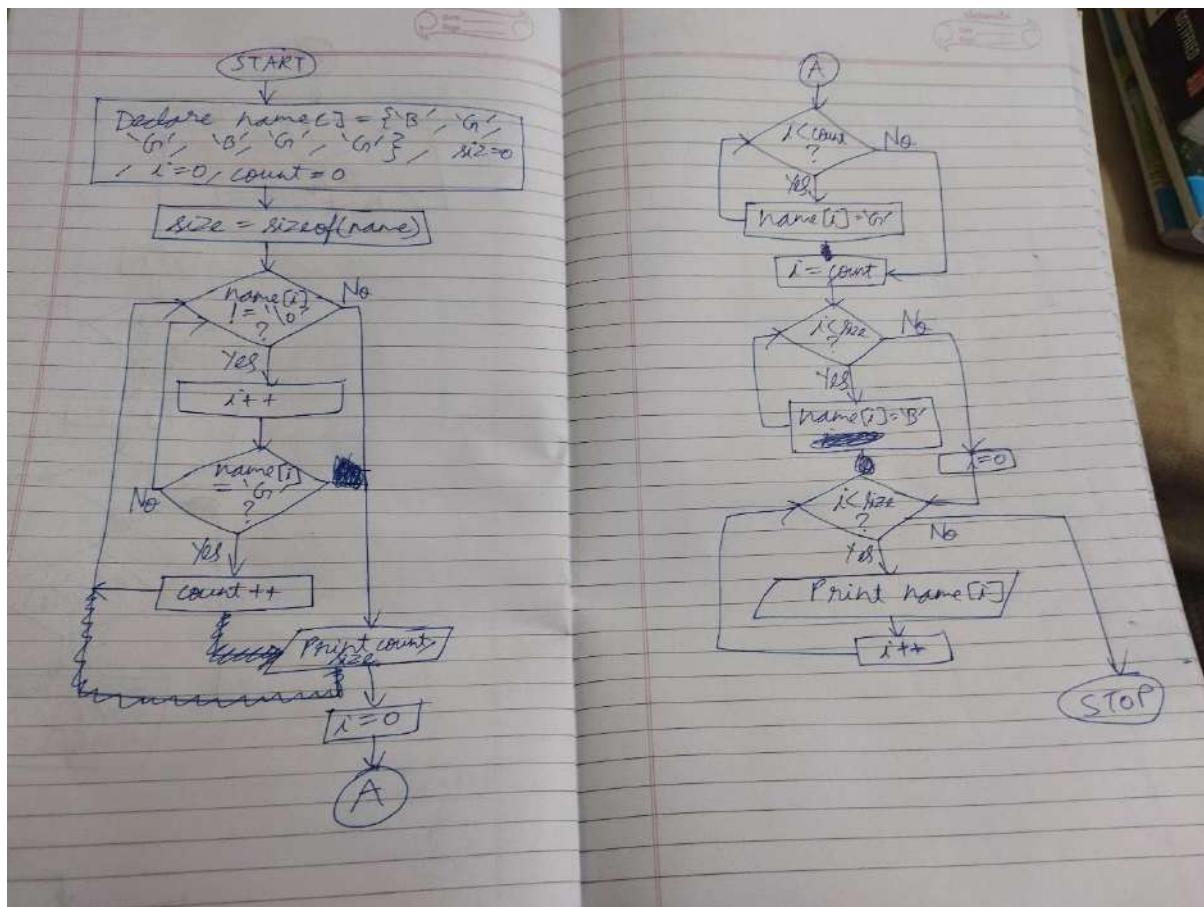
count of girls = 4, size=6
GGGGBB

...Program finished with exit code 0
Press ENTER to exit console.]

```

Inference- We haven't used functions here.

4) 1. START
 2. size \leftarrow sizeof(name)
 3. while name[i] \neq '\0'
 4. i++
 5. If name[i] \neq 'G'
 6. count++
 7. Print count, size
 8. for i < 0 to count do
 9. name[i] = 'G'
 10. for i < count to size do
 11. name[i] = 'B'
 12. i \geq 0
 13. while i < size
 14. Print name[i]
 15. i++
 16. STOP



LAB 12

Aim- Learn to use concept of functions.

1. Write a C program for printing happy numbers up to 50.

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

```
//isHappyNumber() will determine whether a number is  
happy or not
```

```
int isHappyNumber(int num){
```

```
    int rem = 0, sum = 0;
```

```
//Calculates the sum of squares of digits
```

```
    while(num > 0){
```

```
        rem = num%10;
```

```
        sum = sum + (rem*rem);
```

```
        num = num/10;
```

```
}
```

```
    return sum;
```

```
}
```

```
int main()
```

```
{  
//Displays all happy numbers between 1 and 100  
printf("List of happy numbers between 1 and 100: \n");  
for(int i = 1; i< 100; i++){  
int result = i;  
  
//Happy number always ends with 1 and  
//unhappy number ends in a cycle of repeating numbers  
//which contains 4  
while(result != 1 && result != 4){  
result = isHappyNumber(result);  
}  
  
if(result == 1)  
printf("%d\n", i);  
}  
  
return 0;  
}
```

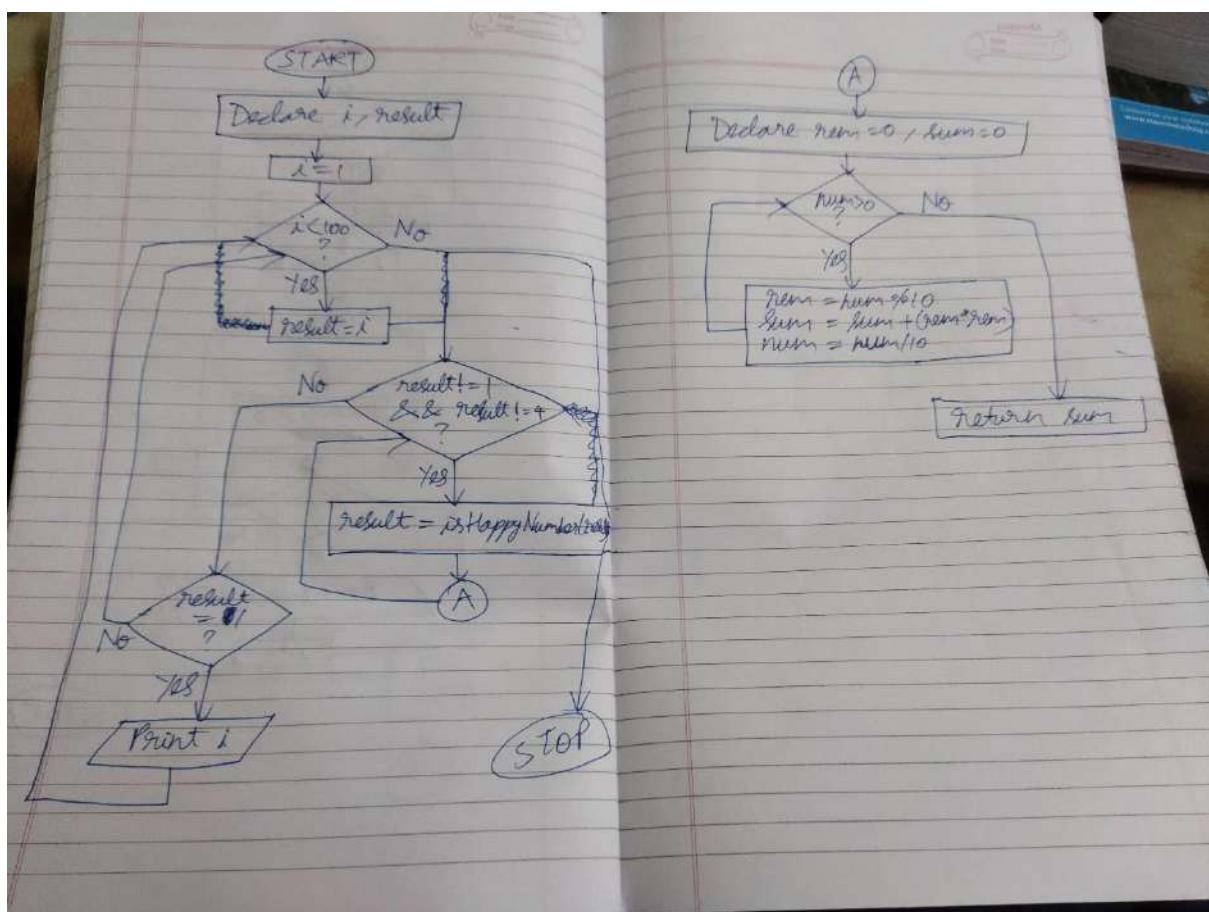
The screenshot shows a terminal window with the following content:

```
main.c
17 t
18 //Displays all happy numbers between 1 and 100
19 printf("List of happy numbers between 1 and 100
20 for(int i = 1; i < 100; i++){
  List of happy numbers between 1 and 100:
1
7
10
13
19
23
28
31
32
44
49
68
70
79
82
86
91
94
97

...
Program finished with exit code 0
Press ENTER to exit console.
```

Inference- We are using functions to print happy numbers between 1 to 100.

1. START
 2. for $i \leftarrow 1$ to 100 do
 3. result $\leftarrow i$
 4. while result $\neq 1$ AND result $\neq 4$
 5. result \leftarrow isHappyNumber(result)
 6. If result ≤ 1
 7. Print i
 8. end for $i \leftarrow 100$
 9. STOP



2. Write a C program for generating Fibonacci-like sequence using Keith Number format.

Input : $x = 197$

Output : Yes

197 has 3 digits, so n = 3

The number is Keith because it appears in the special sequence that has first three terms as 1, 9, 7 and remaining terms evaluated using sum of previous 3 terms.

1, 9, 7, 17, 33, 57, 107, 197,

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

```
int main() {
```

```
    int i, k=3, n;
```

```
    printf("Enter number of terms: ");
```

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

```
    int keith[25];
```

```
    keith[0]=1;
```

```
    keith[1]=9;
```

```
    keith[2]=7;
```

```
    printf("The Keith series is:\n");
```

```
    printf("%d, %d, %d",keith[0],keith[1],keith[2]);
```

```

for (i=3; i<n; i++)
{
    keith[i]= keith[i-1] + keith[i-2]+ keith[i-3];
    printf(", %d", keith[i]);
}

return 0;
}

```

```

main.c
1 #include <stdio.h>
2
3 int main() {
4     int i, n, a = 1, b = 9, c = 7, d;
5
6     printf("Enter number of terms: ");
7     scanf("%d", &n);
8
9     for (i = 1; i < n; i++) {
10        d = a + b;
11        printf("%d, ", d);
12        a = b;
13        b = d;
14    }
15
16    return 0;
17 }

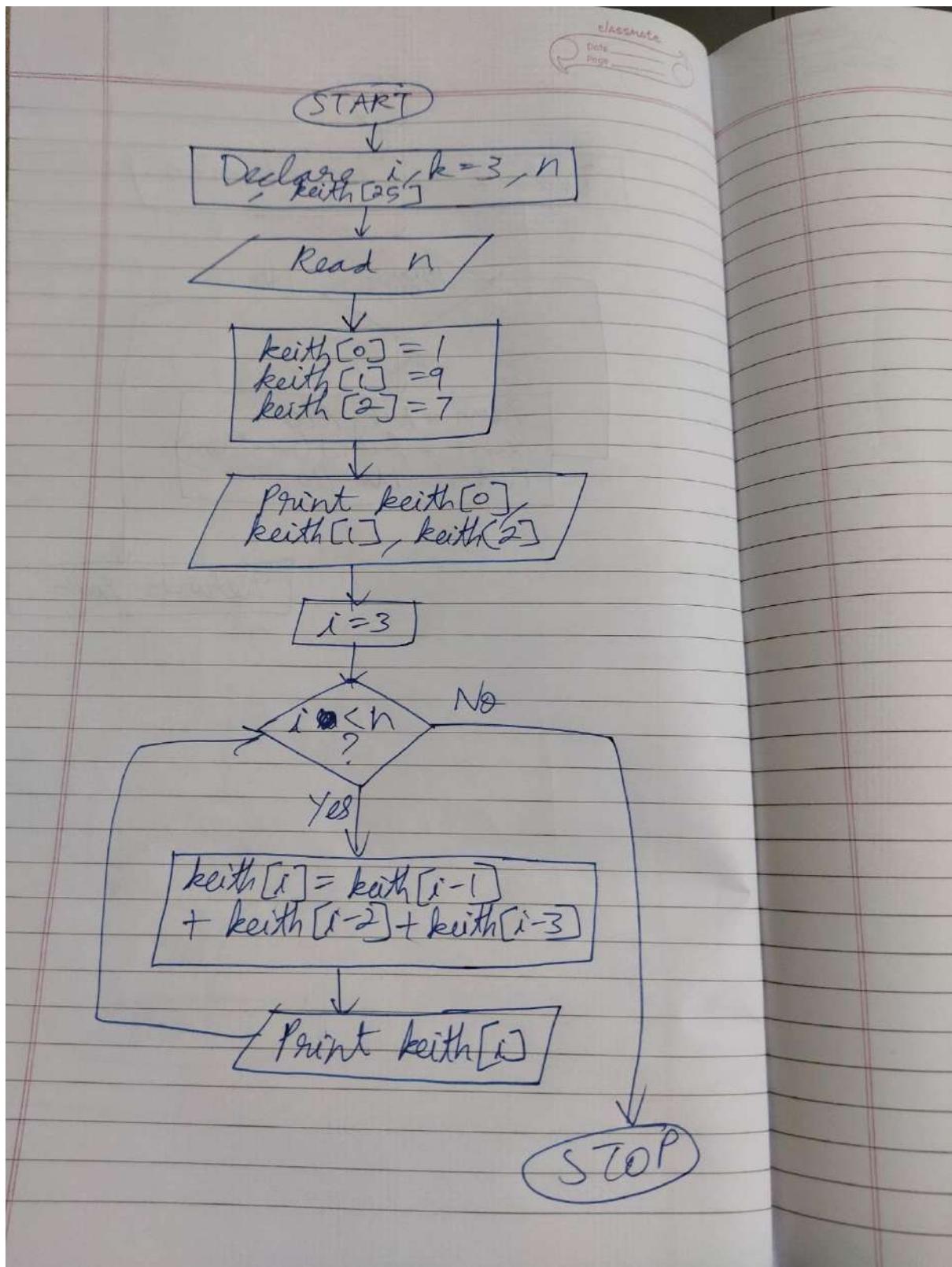
Enter number of terms: 5
The Keith series is:
1, 9, 7, 17, 33

...Program finished with exit code 0
Press ENTER to exit console. []

```

Inference- We are printing Keith series.

- 2) 1. START
2. Read n
3. keith [0] \leftarrow 1
4. keith [1] \leftarrow 9
5. keith [2] \leftarrow 7
6. Print keith[0], keith[1], keith[2]
7. for i \leftarrow 3 to n do
8. keith[i] \leftarrow keith[i-1] + keith[i-2]
 + keith[i-3]
9. Print keith[i]
10. end for i \leftarrow n
11. STOP



3. Write a C program to print prime triplet using functions.

Note: prime triplet in the form of (p, p + 2, p + 6) or (p, p + 4, p + 6).

Output: (5, 7, 11), (7, 11, 13), (11, 13, 17), (13, 17, 19), (17, 19, 23), (37, 41, 43), (41, 43, 47), (67, 71, 73)

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

```
int primetrip(int num){
```

```
    int flag=0, i;
```

```
    for(i=2;i<num;i++){
```

```
        if(num%i==0){
```

```
            flag=1;
```

```
            break;
```

```
        }
```

```
}
```

```
    if(flag==0)
```

```
        return 1;
```

```
    else
```

```
        return 0;
```

```
}
```

```
int main()
{
    int n, N, i, c=0;

    printf("Enter starting number:");
    scanf("%d", &n);

    printf("Enter ending number: ");
    scanf("%d", &N);

    if(n==1){
        n=n+1;
    }

    for(i=n;i<=N;i++){
        if(primetrip(i)==1 && primetrip(i+2)==1 &&
        primetrip(i+6)==1){
            c++;
            printf("%d, %d, %d\n",i,i+2,i+6);
        }
    }
}
```

```

else if(primetrip(i)==1 && primetrip(i+4)==1 &&
primetrip(i+6)==1){

c++;

printf("%d, %d, %d\n",i,i+4,i+6);

}

}

printf("%d is the total number of prime triplets set",c);

return 0;

}

```

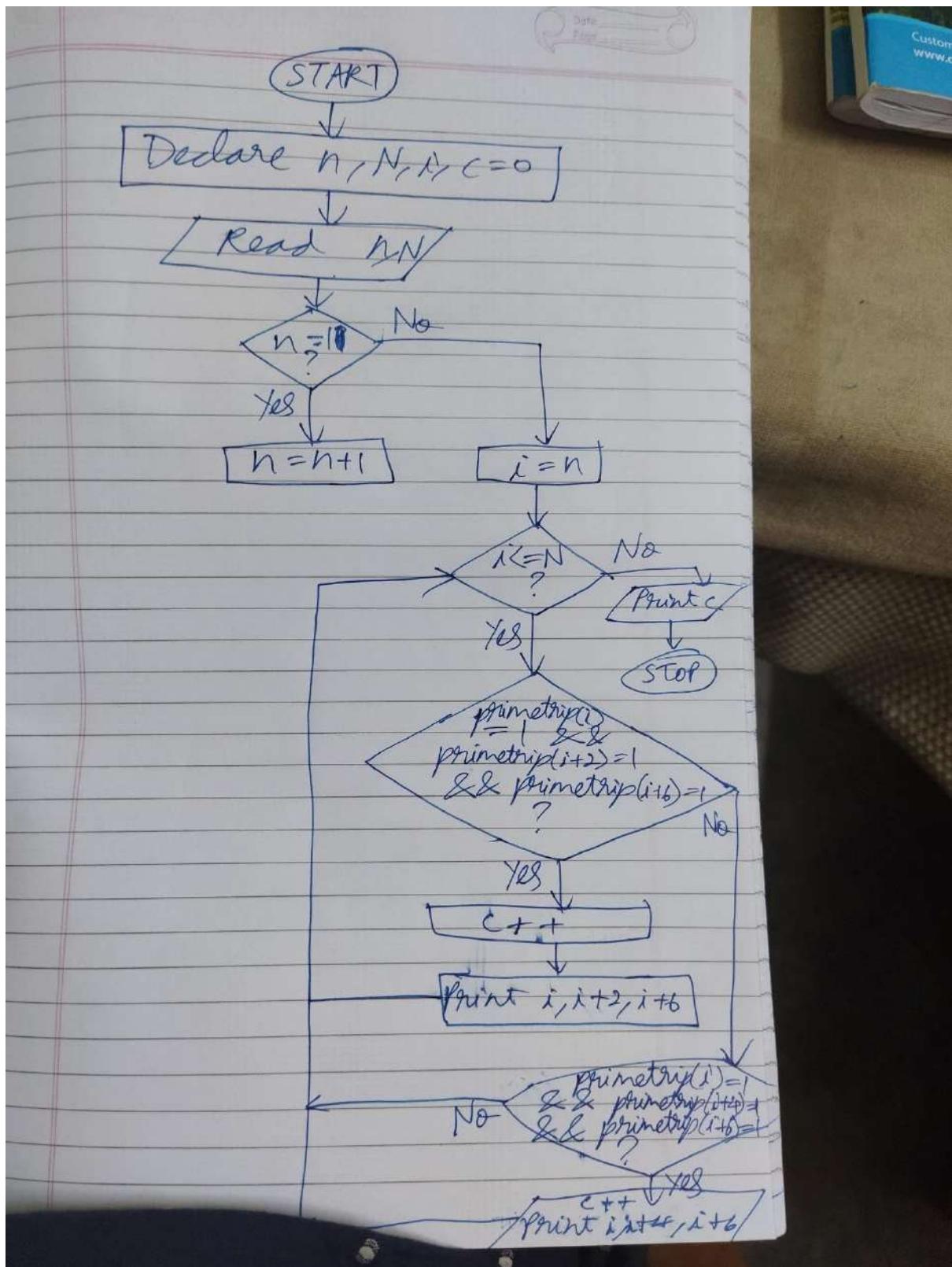
```

main.c
1 #include <stdio.h>
2
3 int primetrip(int num){
4     int flag=0;
5     for(int i=2;i<=sqrt(num);i++)
6         if(num % i==0)
7             flag=1;
8     if(flag==0)
9         return 1;
10    else
11        return 0;
12 }
13
14 int main()
15 {
16     int start, end;
17     printf("Enter starting number:45\n");
18     scanf("%d", &start);
19     printf("Enter ending number: 103\n");
20     scanf("%d", &end);
21
22     for(int i=start;i<end;i++)
23     {
24         if(primetrip(i)==1 && primetrip(i+4)==1 &&
25             primetrip(i+6)==1)
26         {
27             printf("%d, %d, %d\n",i,i+4,i+6);
28             c++;
29         }
30     }
31
32     printf("%d is the total number of prime triplets set\n",c);
33
34     ...Program finished with exit code 0
35 Press ENTER to exit console.█

```

Inference- We are using functions to print prime triplets within the given range input by user.

- 3) 1. START
2. Read n, N
3. If $n \leq 1$
4. $n \leftarrow n + 1$
5. for $i \leq n$ to N do
6. If $\text{primetrip}(i) \leftarrow 1$ &&
 $\text{primetrip}(i+2) \leftarrow 1$ &&
 $\text{primetrip}(i+6) \leftarrow 1$
7. $c++$
8. Print $i, i+2, i+6$
9. Else if ~~$\text{primetrip}(i) \leftarrow 1$~~ &&
 ~~$\text{primetrip}(i+4) \leftarrow 1$~~ &&
 ~~$\text{primetrip}(i+6) \leftarrow 1$~~
10. $c++$
11. Print $i, i+4, i+6$
12. end for $i \leq N$
13. Print c
14. STOP



4. Write a C program to Check whether two strings are anagram of

Samuela Abigail Mathew
71762108039

each other.

Ex: LISTEN- SILENT

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

```
#include <string.h>
```

```
int main () {
```

```
    char s1[25], s2[25], tempstr;
```

```
    int i, j, n, m;
```

```
    printf("Enter string 1:");
```

```
    scanf("%s", s1);
```

```
    printf("Enter string 2:");
```

```
    scanf("%s", s2);
```

```
    n = strlen(s1);
```

```
    m = strlen(s2);
```

```
// If both strings are of different length, then they are not  
anagrams
```

```
if( n != m) {
```

```
    printf("Strings are not anagrams \n");
```

```
return 0;  
}  
  
  
for (i = 0; i < n-1; i++) {  
    for (j = i+1; j < n; j++) {  
        if (s1[i] > s1[j]) {  
            tempstr = s1[i];  
            s1[i] = s1[j];  
            s1[j] = tempstr;  
  
        }  
        if (s2[i] > s2[j]) {  
            tempstr = s2[i];  
            s2[i] = s2[j];  
            s2[j] = tempstr;  
        }  
    }  
}  
  
  
// Compare both strings character by character  
for(i = 0; i<n; i++) {  
    if(s1[i] != s2[i]) {  
        Samuel Abigail Mathew  
        71762108039
```

```

printf("Strings are not anagrams \n");

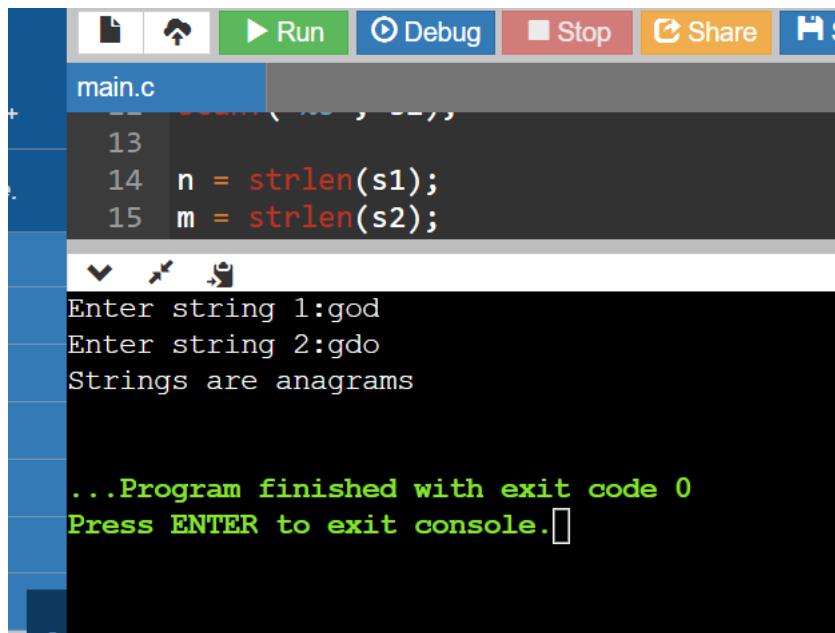
return 0;
}

}

printf("Strings are anagrams \n");

return 0;
}

```



The screenshot shows a code editor interface with a toolbar at the top and a code editor window below. The code editor window displays a file named 'main.c' with the following content:

```

main.c
13
14 n = strlen(s1);
15 m = strlen(s2);

Enter string 1:god
Enter string 2:gdo
Strings are anagrams

...Program finished with exit code 0
Press ENTER to exit console. []

```

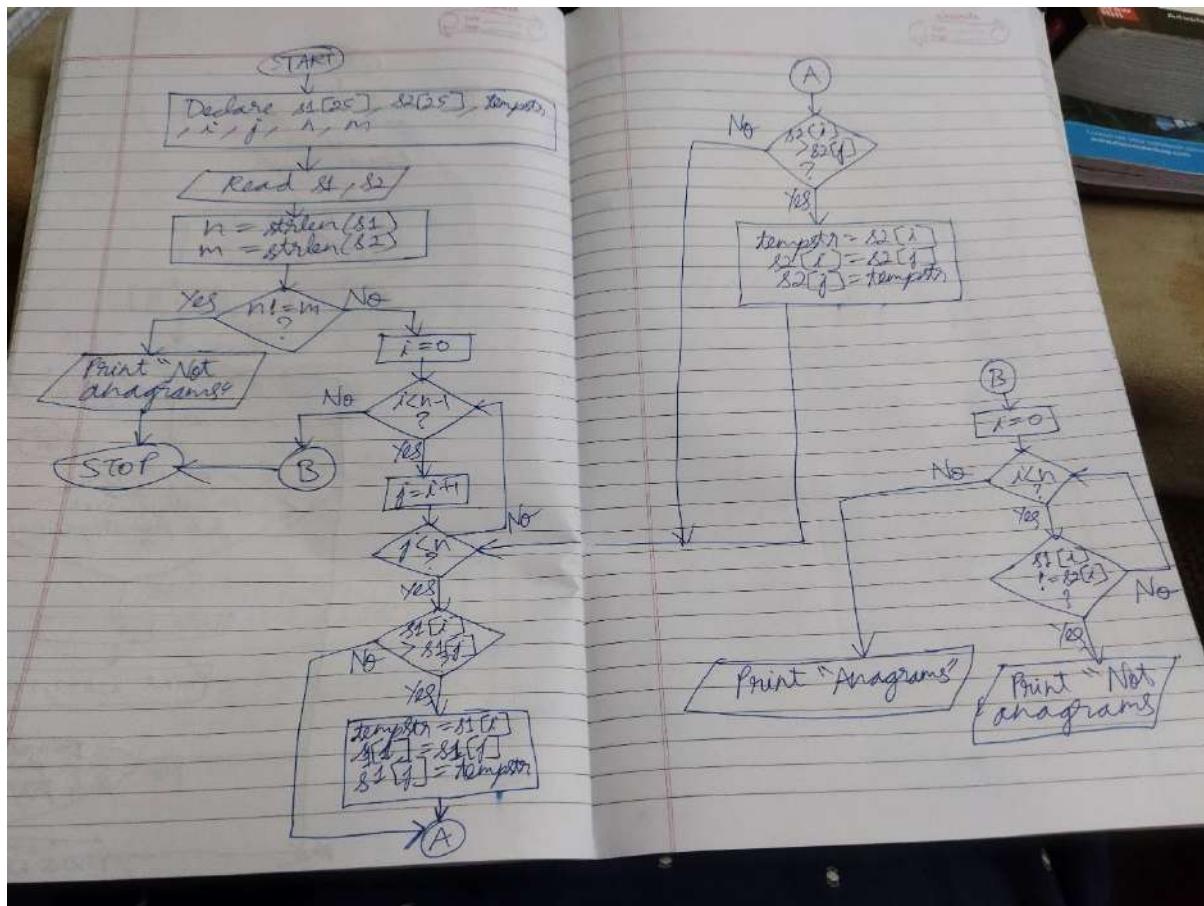
The code uses the `strlen` function to determine the lengths of the input strings `s1` and `s2`. It then compares the lengths. If they are equal, it proceeds to sort the characters of both strings and compare them character by character to determine if they are anagrams. The output window shows the user input 'god' and 'gdo', and the program's response 'Strings are anagrams'. The terminal also displays the completion message '...Program finished with exit code 0'.

Inference- two strings are taken as input and then they are checked for string length. If both have same length then the program further checks by arranging the characters of the string in ascending order and checking each character of the string. This is how the program checks if the strings are anagrams or not.

is not palindrom

2) 1. START
 2. Read
 3. $n \leftarrow \text{strlen}(s_1)$
 4. $m \leftarrow \text{strlen}(s_2)$
 5. If $n \neq m$
 6. Print "Not anagrams"
 7. Exit
 8. for $i \leftarrow 0$ to $n-1$ do
 9. for $j \leftarrow i+1$ to n do
 10. If $s_1[i] > s_1[j]$
 11. $\text{tempstr} \leftarrow s_1[i]$
 12. $s_1[i] \leftarrow s_1[j]$
 13. $s_1[j] \leftarrow \text{tempstr}$
 14. If $s_2[i] > s_2[j]$
 15. $\text{tempstr} \leftarrow s_2[i]$
 16. $s_2[i] \leftarrow s_2[j]$
 17. $s_2[j] \leftarrow \text{tempstr}$
 18. end for $j \leftarrow n$
 19. end for $i \leftarrow n-1$
 20. for $i \leftarrow 0$ to n do

21. If $s1[i] \neq s2[i]$
 22. Print ("Not anagrams")
 23. Exit
 24. end for $i \leq n$
 25. Print "Anagrams"
 26. STOP



5. Write a C program to find minimum occurring character in a string.

#include <stdio.h>

Samuela Abigail Mathew
71762108039

```
#define MAX_SIZE 100 // Maximum string size
#define MAX_CHARS 255 // Maximum characters allowed

int main()
{
    char str[MAX_SIZE];
    int freq[MAX_CHARS]; //Stores frequency of each character
    int i = 0, min;
    int ascii;

    printf("Enter any string: ");
    scanf("%s",str);

    /* Initialize frequency of all characters to 0 */
    for(i=0; i<MAX_CHARS; i++)
    {
        freq[i] = 0;
    }

    /* Finds frequency of each characters */
    i=0;
    while(str[i] != '\0')
```

```
{  
    ascii = (int)str[i];  
    freq[ascii] += 1;  
  
    i++;  
}  
  
/* Finds minimum frequency */  
min = 0;  
for(i=0; i<MAX_CHARS; i++)  
{  
    if(freq[i] != 0)  
  
    {  
        if(freq[min] == 0 || freq[i] < freq[min])  
            min = i;  
    }  
}  
  
printf("Minimum occurring character is %c = %d times", min,  
freq[min]);
```

```
return 0;  
}
```

The screenshot shows a code editor interface with a menu bar at the top. The menu bar includes icons for file operations (New, Open, Save), and buttons for Run, Debug, Stop, Share, and Save. Below the menu bar, the file name "main.c" is displayed. The code in the editor is:

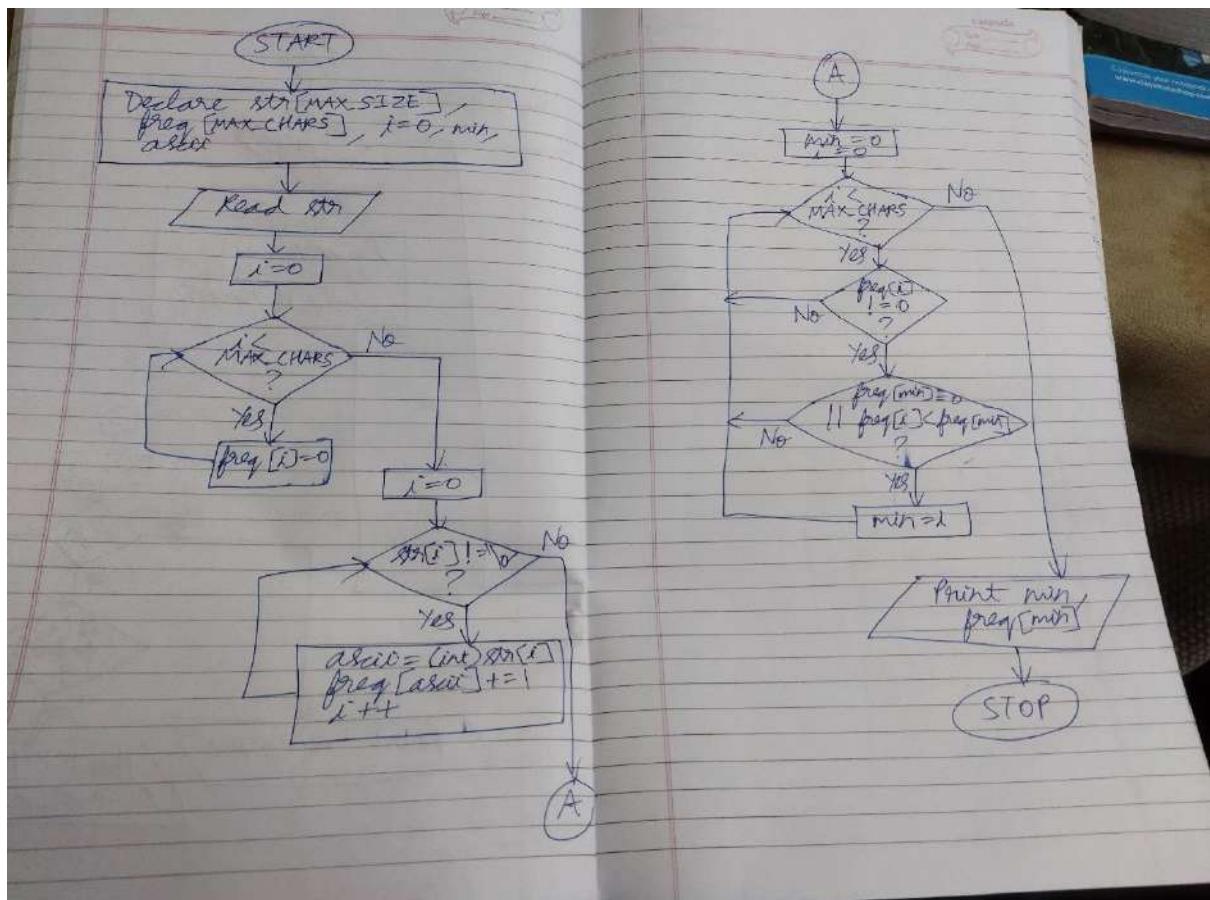
```
28 ,  
29 }  
30  
31 /* Finds minimum frequency */
```

Below the code editor is a terminal window showing the program's execution. The terminal output is:

```
Enter any string: hello sam  
Minimum occurring character is e = 1 times  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Inference-We are using functions to find character having minimum frequency, but this is not accurate.

5) 1. START
 2. Read str
 3. for $i \leftarrow 0$ to MAX_CHARS do
 4. freq[i] = 0
 5. end for $i \leftarrow \text{MAX_CHARS}$
 6. $i \leftarrow 0$
 7. while $\text{str}[i] \neq '\text{O}'$
 8. ascii $\leftarrow (\text{int})\text{str}[i]$
 9. freq[ascii] += 1
 10. $i++$
 11. min $\leftarrow 0$
 12. for $i \leftarrow 0$ to MAX_CHARS do
 13. If freq[i] == 0
 14. If freq[min] == 0 OR
 freq[i] < freq[min]
 min $\leftarrow i$
 15. end for $i \leftarrow \text{MAX_CHARS}$
 16. Print min, freq[min]
 17. STOP



6. Write a C program to delete all duplicate elements from an array using functions.

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

void dupremove(int len, char str[]){
    for(int i=0; i<len; i++){
        char ch = str[i];
        //Check if current character matches any other character in
        //the subsequence
        for(int j=i+1; j<len; ){
            if(str[i] == str[j]){
                //If yes, then shift the right characters to the left
                for(int k=j; k<len; k++){
                    str[k] = str[k+1];
                }
                len--;
            }
        }
    }
}
```

```
        else {
            //only increment if the duplicate is not found
            //because after the shift, index j can again have
            duplicate
            j++;
        }
    }
}

printf(" %s",str);
}

int main()
{
    char string[30];
    printf("Enter string: ");
    scanf("%s",string);
    int length= strlen(string);

    printf("String after removing all duplicates is" );
    dupremove(length, string);

    return 0;
}
```

The screenshot shows a C IDE interface with the following elements:

- Toolbar:** Includes icons for File, Upload, Run, Debug, Stop, Share, Save, and Help.
- Code Editor:** The file `main.c` is open. The code is as follows:

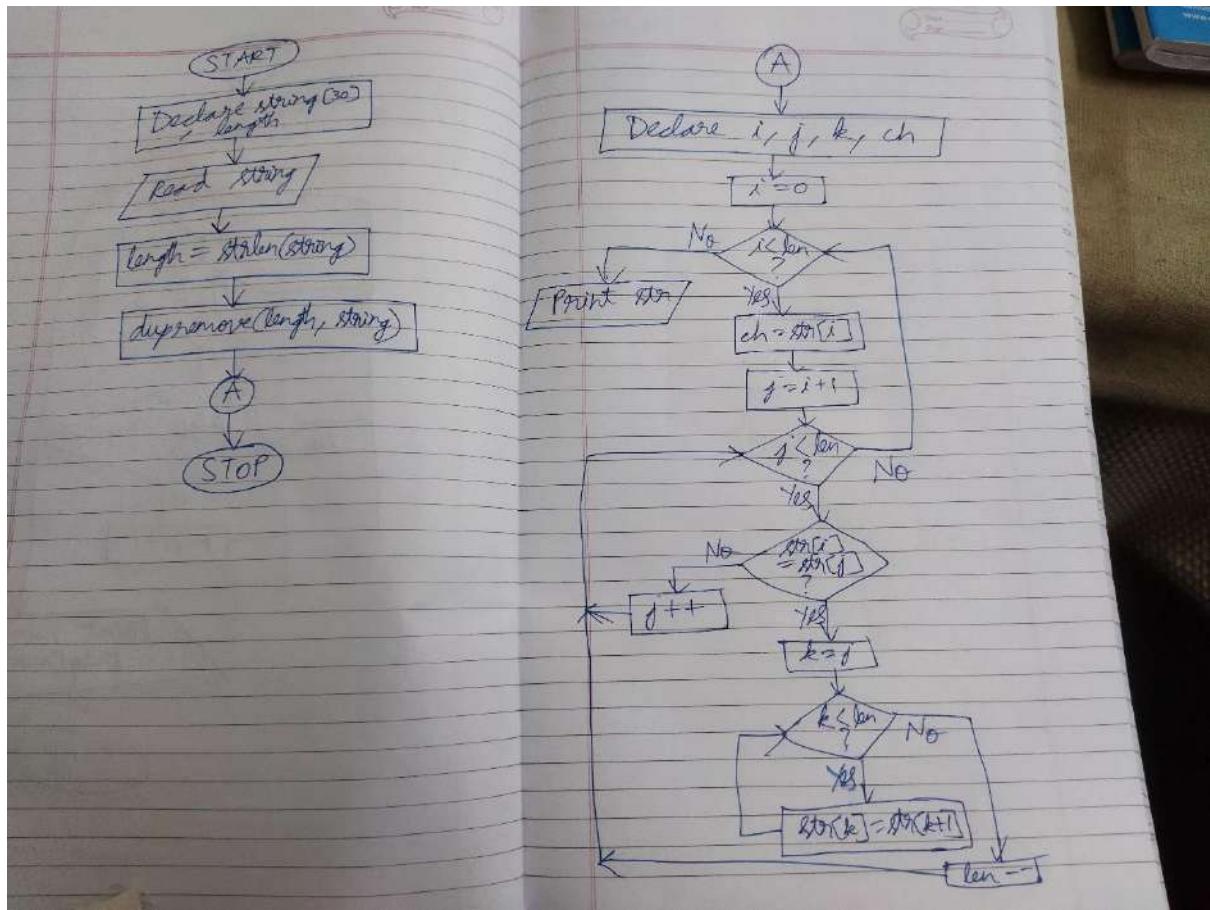
```
10
11     if(str[i] == str[j]){
12
13         //If yes, then shift the right character
```
- Output Console:** Displays the following text:

```
Enter string: samuelaabigail
String after removing all duplicates is samuelbig

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- When duplicate is found, we shift position of remaining characters towards the left inside function.

- 6.) 1. START
2. Read string
3. length \leftarrow strlen(string)
4. call dupremove(length, string)
5. for $i \leftarrow 0$ to len do
6. ch \leftarrow str[i]
7. for $j \leftarrow i+1$ to len do
8. If str[i] \leftarrow str[j]
9. for $k \leftarrow j$ to len do
10. str[k] \leftarrow str[k+1]
11. end for k \leftarrow len
12. len - -
13. Else
14. j ++
15. end for j \leftarrow len
16. end for i \leftarrow len
17. Print str
18. STOP



LAB 13

Aim- Learn to use concept of functions and pointers.

1. Given a string “str” of size “n”. Check whether it’s a palindrome or not.

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

```
// Function to check if the string is palindrome
```

```
// using pointers
```

```
void isPalindrome(char* string)
```

```
{
```

```
char *ptr, *rev;
```

```
ptr = string;
```

```
while (*ptr != '\0') {
```

```
    ++ptr;
```

```
}
```

```
--ptr;
```

```
for (rev = string; ptr >= rev;) {
```

```
if (*ptr == *rev) {  
    --ptr;  
    rev++;  
}  
else  
break;  
}  
  
  
if (rev > ptr)  
printf("String is Palindrome");  
else  
printf("String is not a Palindrome");  
}  
  
  
// Driver code  
int main()  
{  
char str[100] = "madam";  
  
  
isPalindrome(str);  
  
  
return 0;
```

}

The screenshot shows a code editor window with a dark theme. At the top, there is a toolbar with icons for file operations (New, Open, Save), Run, Debug, Stop, Share, and Save. The file tab shows "main.c". The code in the editor is:

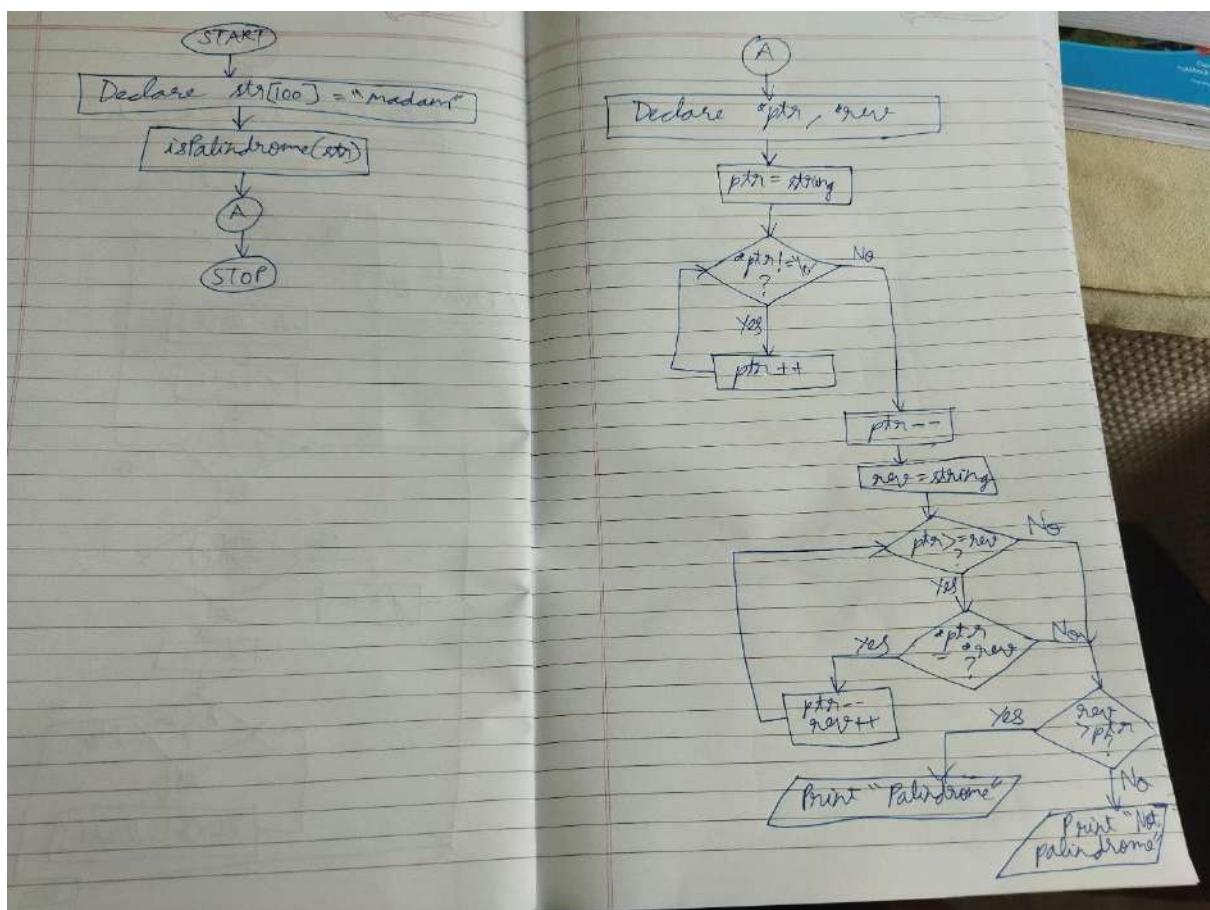
```
18 }
19 else
20 break;
21 }
22
23 if (rev > ptr)
24 printf("String is Palindrome");
25 else
26 printf("String is not a Palindrome");
27 }
28
29 // Driver code
30 int main()
31 {
32     char str[1000] = "madam";
33
34     isPalindrome(str);
35
36     return 0;
37 }
38
39 }
```

Below the code editor is a terminal window showing the output of the program. The output is:

```
String is Palindrome
...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- We are checking if string is palindrome or not using function and pointers.

1. START
 2. Declare str[100] = "madam"
 3. Call isPalindrome(str)
 4. Declare *ptr, *rev
 5. ptr = string
 6. for *ptr! = '\0' do
 7. ptr++
 8. for rev = string to ptr do
 9. If *ptr < *rev
 10. ptr--
 11. rev++
 12. Else
 13. break
 14. end for rev > ptr
 15. If rev > ptr
 16. Print "Palindrome"
 17. Else
 18. Print "Not palindrome"
 19. STOP



2. Copy a source string into destination string.

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

void copy_string(char*, char*);
int main()
{
    char source[100], target[100];
    printf("Enter source string\n");
    scanf("%s",source);
    copy_string(target, source);
    printf("Target string is \"%s\"\n", target);
    return 0;
}

void copy_string(char *target, char *source)
{
    while(*source)
    {
        *target = *source;
        source++;
        target++;
    }
}
```

```
}
```

```
*target = '\0';
```

```
}
```

The screenshot shows a C IDE interface. At the top, there's a toolbar with icons for file operations (New, Open, Save), Run, Debug, Stop, Share, and Help. Below the toolbar, the file name "main.c" is displayed. The code editor shows the following code:

```
main.c
6 {
7     char source[100], target[100];
8     printf("Enter source string\n");
```

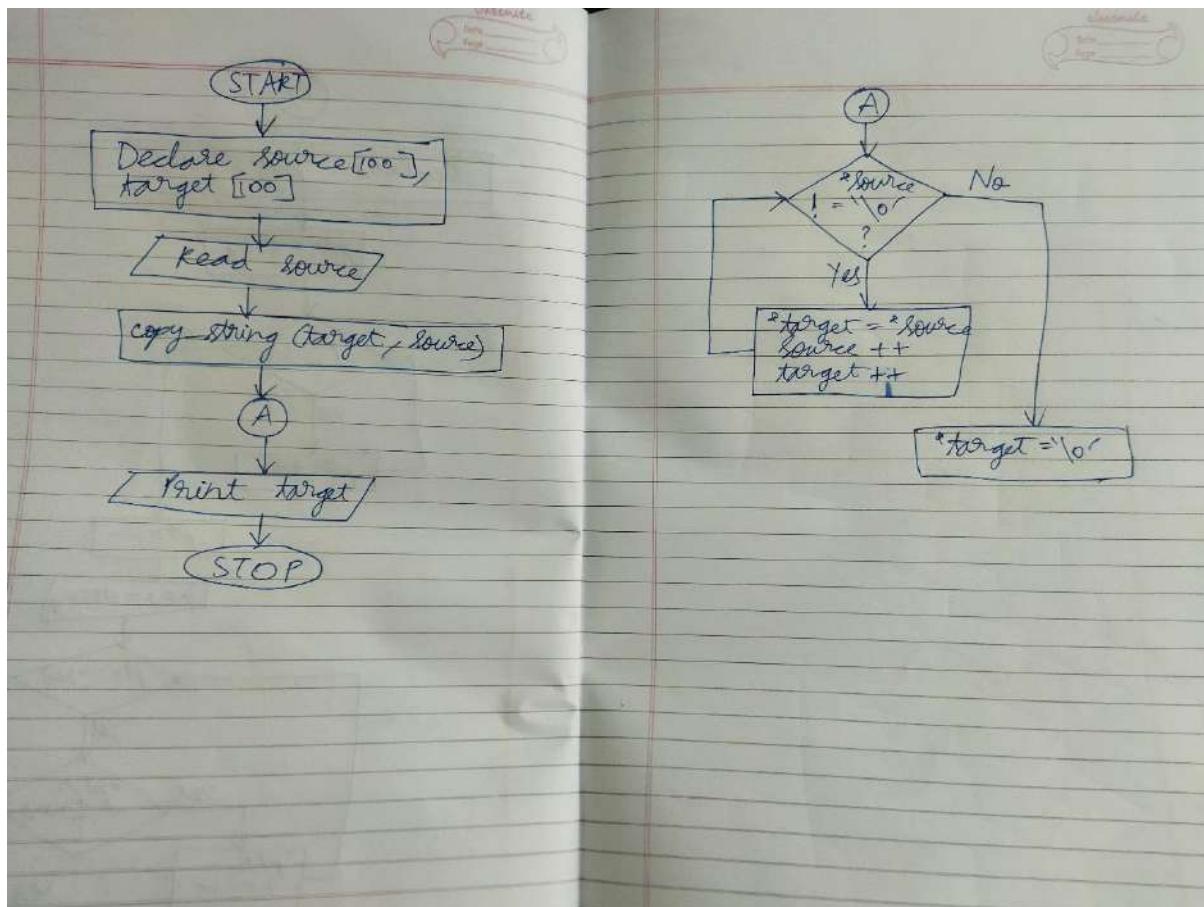
Below the code editor is a terminal window showing the program's output:

```
Enter source string
hello
Target string is "hello"

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- We are copying string character by character using pointers.

1. START
 2. Declare str[100] = "madam"
 3. Call isPalindrome(str)
 4. Declare *ptr, *rev
 5. ptr = string
 6. for *ptr != '\0' do
 7. ptr++
 8. for rev = string to ptr do
 9. If *ptr == *rev
 10. ptr
 11. rev++
 12. Else
 13. break
 14. end for rev > ptr
 15. If rev > ptr
 16. Print "Palindrome"
 17. Else
 18. Print "Not palindrome"
 19. STOP



3. Insert a string “J” into string “I” at position “pos”.

```
#include <string.h>
#include <stdlib.h>
void insert_substring(char*, char*, int);
char* substring(char*, int, int);

int main()
{
    char text[100], substring[100];
    int position;

    printf("Enter some text\n");
    scanf("%s",text);

    printf("Enter a string to insert\n");
    scanf("%s",substring);

    printf("Enter the position to insert\n");
    scanf("%d", &position);

    insert_substring(text, substring, position);
```

```
printf("%s\n",text);  
return 0;  
}
```

```
void insert_substring(char *a, char *b, int position)
```

```
{  
char *f, *e;  
int length;
```

```
length = strlen(a);
```

```
f = substring(a, 1, position - 1 );
```

```
e = substring(a, position, length-position+1);
```

```
strcpy(a, "");  
strcat(a, f);  
free(f);  
strcat(a, b);  
strcat(a, e);  
free(e);  
}
```

Samuela Abigail Mathew
71762108039

```
char *substring(char *string, int position, int length)
{
    char *pointer;

    int c;

    pointer = malloc(length+1);

    if( pointer == NULL )
        exit(EXIT_FAILURE);

    for( c = 0 ; c < length ; c++ )
        *(pointer+c) = *((string+position-1)+c);

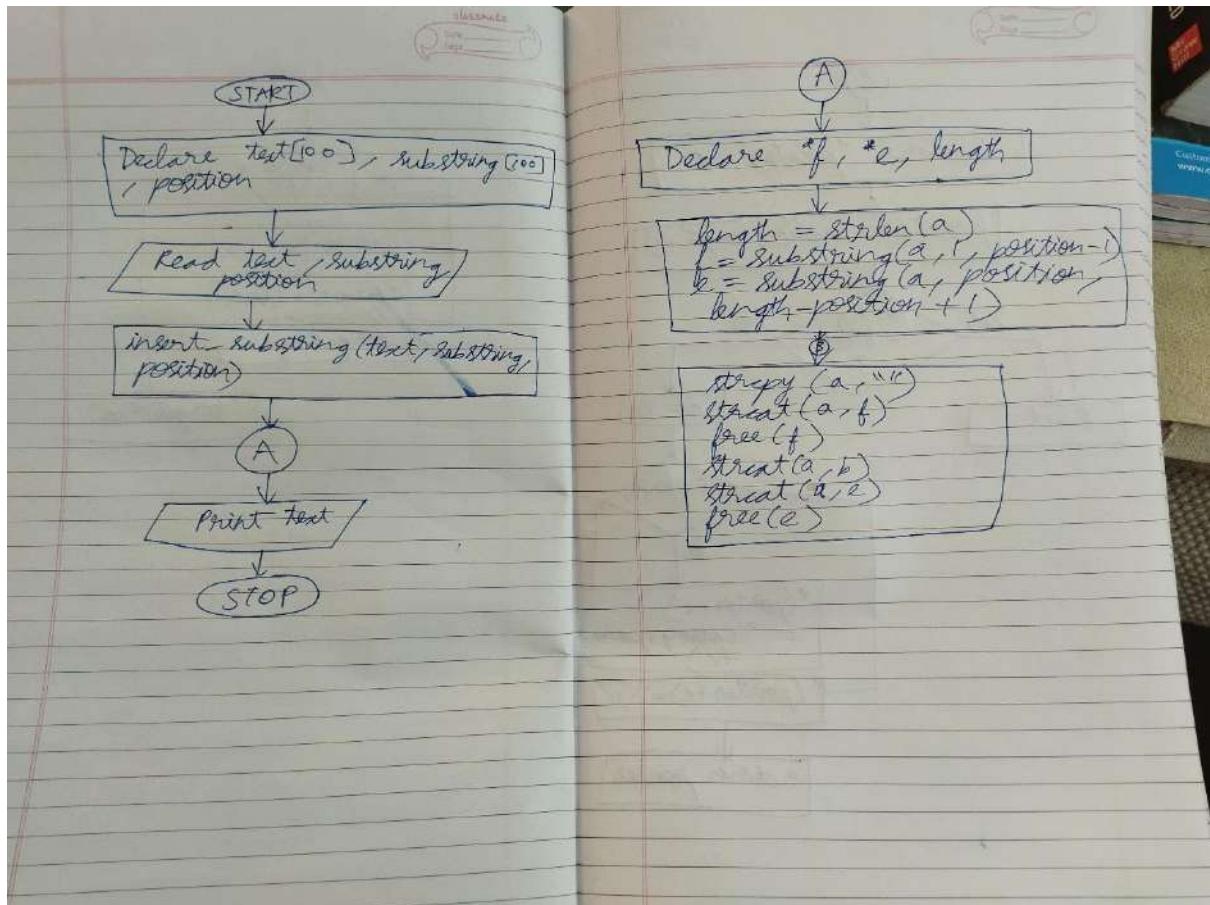
    *(pointer+c) = '\0';

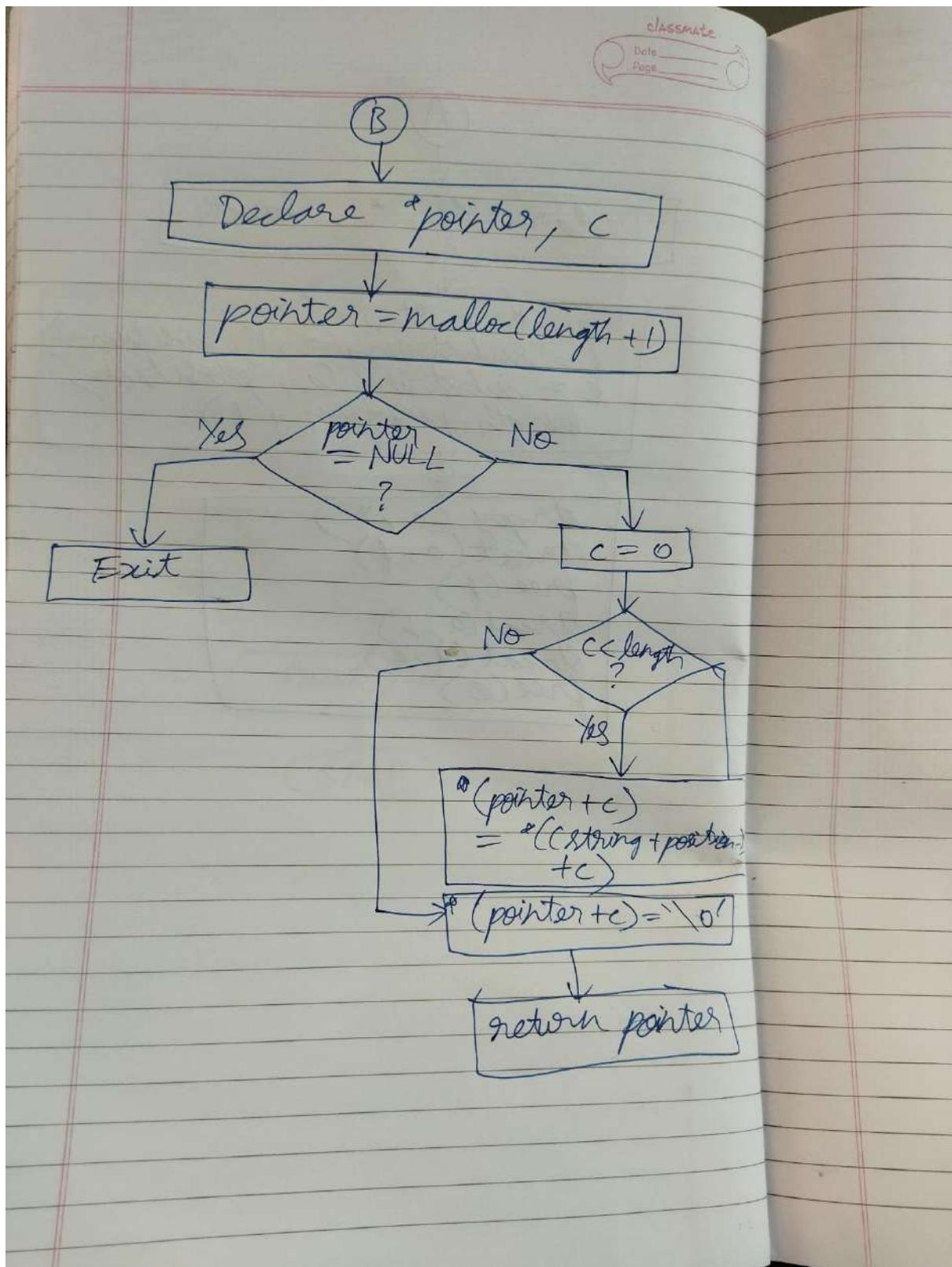
    return pointer;
}
```

```
main.c
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4
5 Enter some text
6 samula
7 Enter a string to insert
8 e
9 Enter the position to insert
10 5
11 samuela
12
13 ...Program finished with exit code 0
14 Press ENTER to exit console.
```

Inference- After inserting string in given position, we are shifting remaining characters towards right.

- Date _____
Page _____
1. START
 2. Declare text[100], substring[100]
position
 3. Read text, substring, position
 4. insert_substring(text, substring,
position)
 5. Declare *f, *l, length
 6. length ← strlen(a)
 7. f ← substring(a, 1, position - 1)
 8. l ← substring(a, position,
length - position + 1)
 9. Declare *pointer, c
 10. pointer ← malloc(length + 1)
 11. If pointer < NULL
exit(0)
 12. for (c <= 0) to length do
 13. *(pointer + c) = *((string
+ position - 1) + c)
 14. end for c ← length
 15. *(pointer + c) ← '\0'
 16. return pointer
 17. strcpy(a, "")
 18. strcat(a, f)
 19. free(f)
 20. strcat(a, b)
 21. strcat(a, e)
 22. free(e)
 23. Print text
 24. STOP





LAB 14

Aim- Learn to use concept of functions and pointers.

Implement C programs for the following problem statements:

1. Find the Length of String using Pointers.

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

```
#include<string.h>
```

```
int string_ln(char*);
```

```
void main() {
```

```
    char str[20];
```

```
    int length;
```

```
    printf("\nEnter any string : ");
```

```
    gets(str);
```

```
    length = string_ln(str);
```

```
    printf("The length of the given string %s is : %d", str, length);
```

```
}
```

```

int string_ln(char*p) /* p= &str[0] */

{
    int count = 0;

    while (*p != '\0') {

        count++;

        p++;
    }

    return count;
}

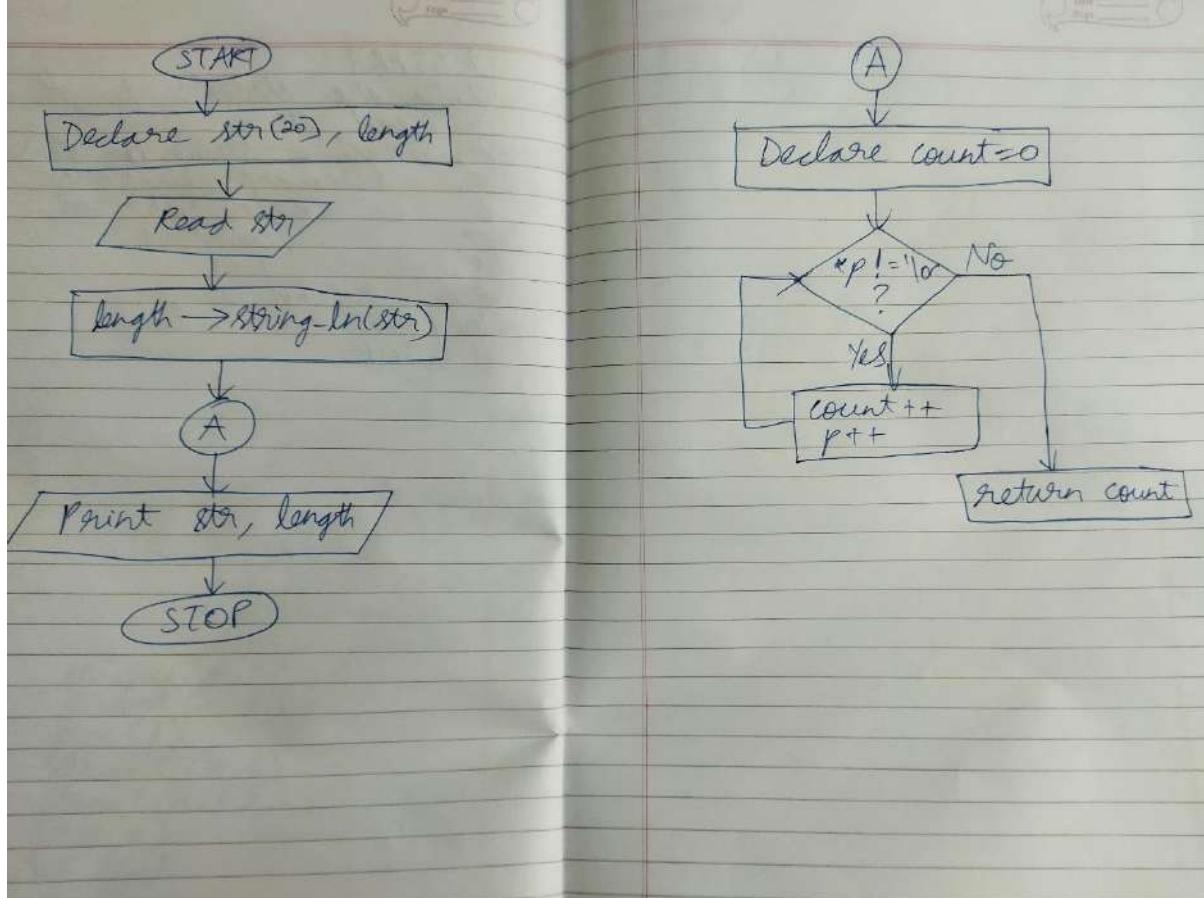
```

```

main.c
  6 void main()
  7 char str[20];
  8 int length;
  9
  10
  11 | gets(str);
  12 | ^~~~                                         input
main.c:11:1: warning: 'gets' is deprecated [-Wdeprecated-declarations]
  11 | gets(str);
  12 | ^~~~                                         In file included from main.c:1:
/usr/include/stdio.h:577:14: note: declared here
  577 | extern char *gets (char *__s) __wur __attribute_deprecated__;
  578 | ^~~~                                         /usr/bin/ld: /tmp/ccMYBqb0.o: in function `main':
main.c:(.text+0x34): warning: the `gets' function is dangerous and should not be used.
  580 |
  581 < Enter any string : samuela abigail
  582 The length of the given string samuela abigail is : 15
  583 ...
...Program finished with exit code 0
Press ENTER to exit console.[]
```

Inference- While character of string is not equal to null, it'll keep incrementing count by 1. Final value of count is length of string.

1. START
2. Declare str[20], length
3. Read str
4. length \leftarrow string.length(str)
5. Declare count $\leftarrow 0$
6. while *p != '0'
7. count ++
8. p ++
9. return count
10. Print str, length
11. STOP



2. To read array of elements and print the value with the addresses.

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

```
int main()
{
    int arr[10]; //declare integer array
    int *pa; //declare an integer pointer
    int i;
```

```
    pa=&arr[0]; //assign base address of array
```

```
    printf("Enter 5 array elements:\n");
    for(i=0;i < 5; i++){
        printf("Enter element %02d: ",i+1);
        scanf("%d",pa+i); //reading through pointer
    }
```

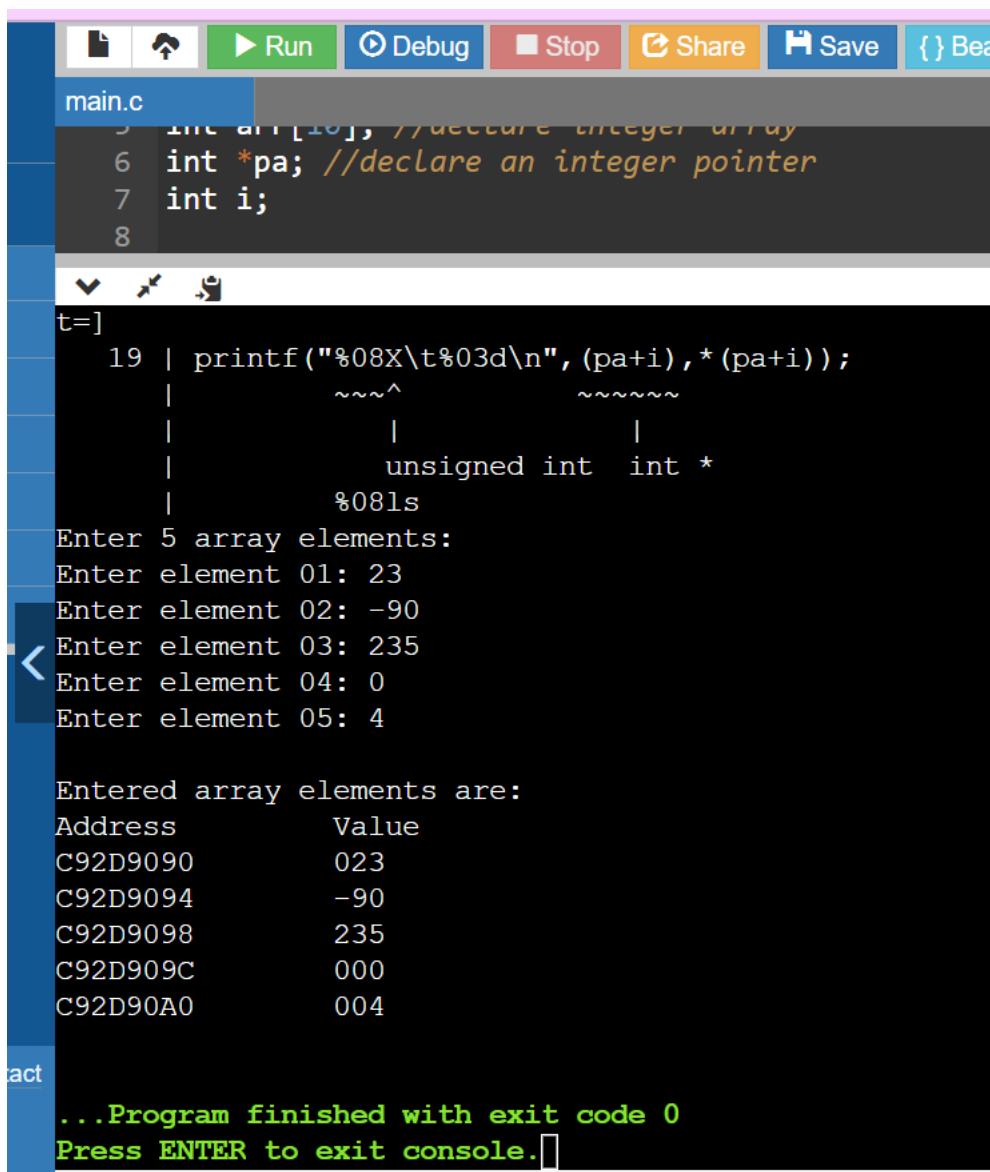
```
    printf("\nEnterd array elements are:");
```

```
    printf("\nAddress\tValue\n");
```

```
    for(i=0;i<5;i++){
        printf("%08X\t%03d\n",*(pa+i),*(pa+i));
    }
```

Samuela Abigail Mathew
71762108039

```
return 0;  
}  
  
}
```



The screenshot shows a C IDE interface with the following details:

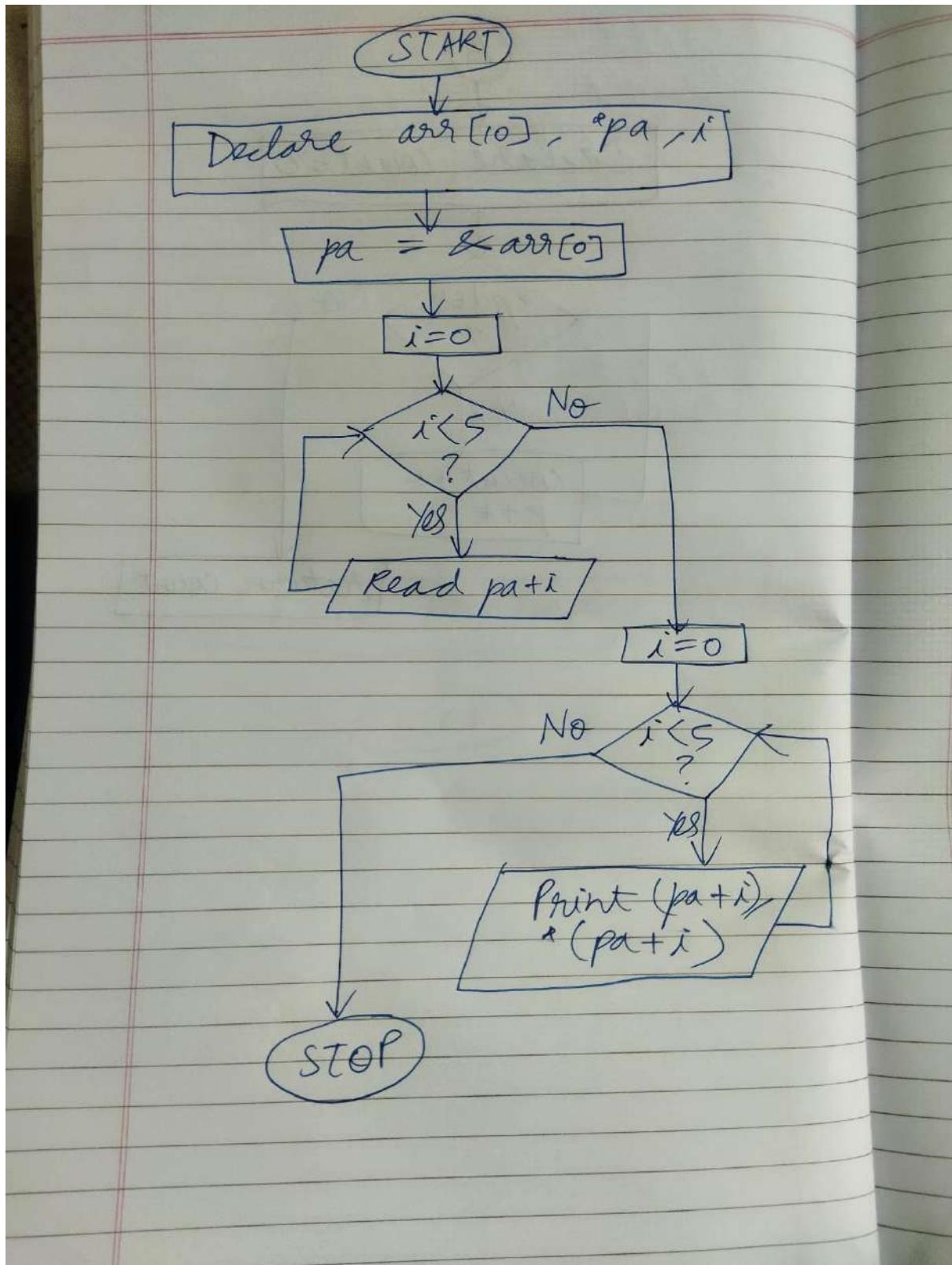
- Toolbar:** Run, Debug, Stop, Share, Save, Bea
- Code Editor:** File main.c contains:

```
int arr[10], //declare integer array  
6 int *pa; //declare an integer pointer  
7 int i;  
8  
t=]  
19 | printf("%08X\t%03d\n", (pa+i),*(pa+i));  
|~~~~~^~~~~~  
| |  
| unsigned int int *  
| %08ls  
Enter 5 array elements:  
Enter element 01: 23  
Enter element 02: -90  
Enter element 03: 235  
Enter element 04: 0  
Enter element 05: 4  
  
Entered array elements are:  
Address Value  
C92D9090 023  
C92D9094 -90  
C92D9098 235  
C92D909C 000  
C92D90A0 004
```
- Output Window:** Shows the program's execution:

```
...Program finished with exit code 0  
Press ENTER to exit console.
```

Inference- We are printing address in hexadecimal format.

1. START
2. Declare arr[10], *pa, i
3. pa \leftarrow &arr[0]
4. for i \leftarrow 0 to 5 do
5. Read pa+i
6. end for i \leq 5
7. for i \leftarrow 0 to 5 do
8. Print (pa + i), *(pa + i)
9. end for i \leq 5
10. STOP



3. First string from the given array whose reverse is also present in the same array

Input: str[] = {"hi", "geeks", "apple", "for", "elppa", "skeeg"}
(geeks is first string having reverse string also)

Output: geeks

```

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

void reverse(char *str) {
    int len, i;
    char *start, *end, temp;

    len = strlen(str);
    start = str;
    end = str; } 0th position's value (for hi, it'll be h)

    for (i=0; i<len-1; i++) { end now has value of
        end++; } end's 0th position (h)

    for (i=0; i<len/2; i++) { (for "hi", len=2. So i<1, so i=0. When i=0,
        temp = *end; i+1, so i=1. When i=1, end=i (in "hi")
        *end = *start; Swapping to get reverse
        *start = temp;
        start++;
        end--;
    }
}

int strlen(char *ptr) {
    int i=0;
    while (*ptr+i != '\0') { Finding length of string
        i++;
    }
    return i;
}

```

Output: First occurrence of reverse string
is apple whose reverse is elppa
At position 5

```
int main() {
    int i, j, flag = 0;
    char str[10][10] = {"hi", "apple",
                        "geeks", "for", "elppa", "sheeg"};
    char dest[15] = "", reverse[15], temp;
    char (*ptrArr)[10] = NULL;

    ptrArr = str;

    for (i = 0; i < 5; i++) {
        if (flag != 0)
            break;

        strcpy(dest, &ptrArr[i]);
        reverse(dest);

        for (j = 0; j < 5; j++) {
            if (strcmp(dest, &ptrArr[j]) == 0) {
                flag = 1;
                printf("First occurrence
of reverse string: %s\n At position
%d", reverse);
                strcpy(temp, ptrArr[j]);
                reverse(temp);
                strcpy(reverse, temp);

                printf("First occurrence of
reverse string is %s whole reverse
is %s\n At position %d", reverse,
&ptrArr[j], (j + 1));
                break;
            }
        }
    }

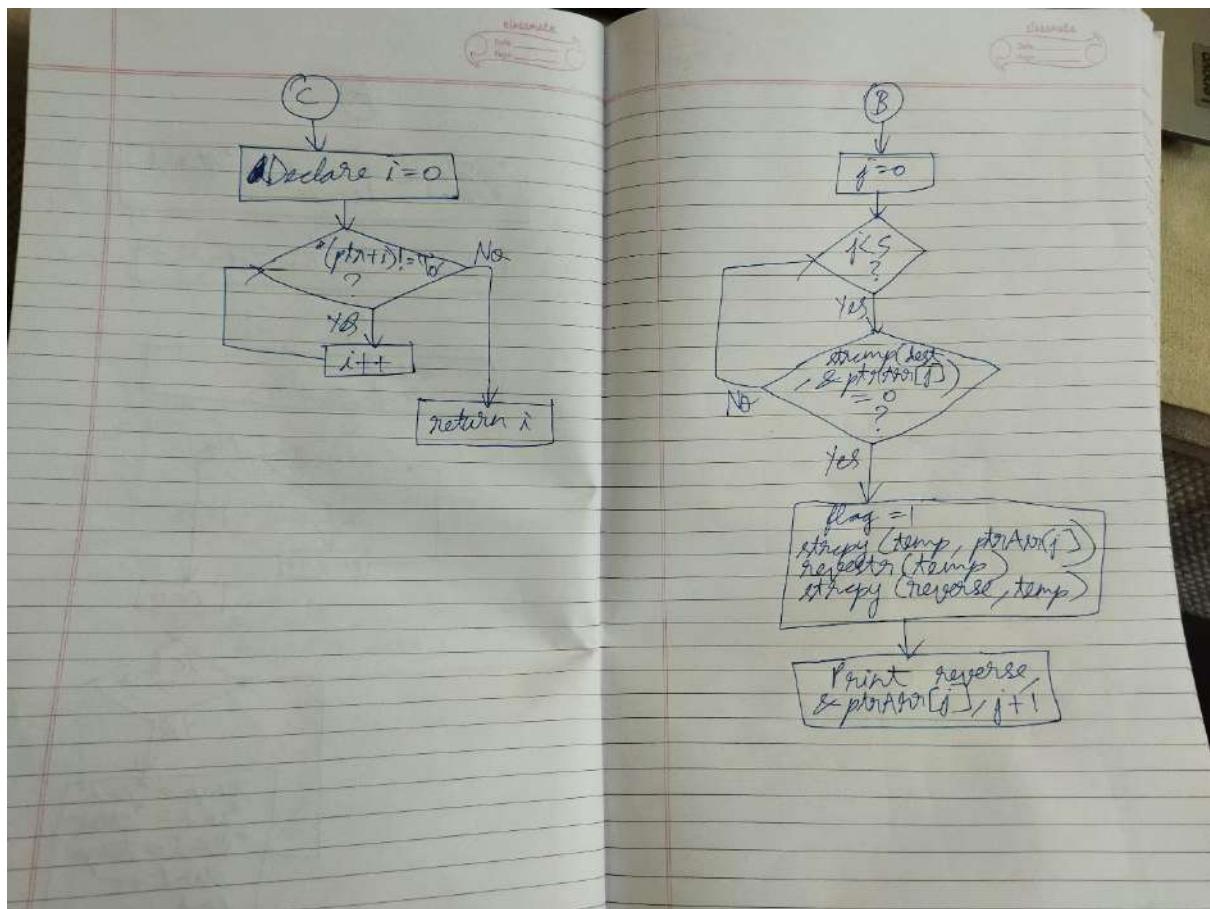
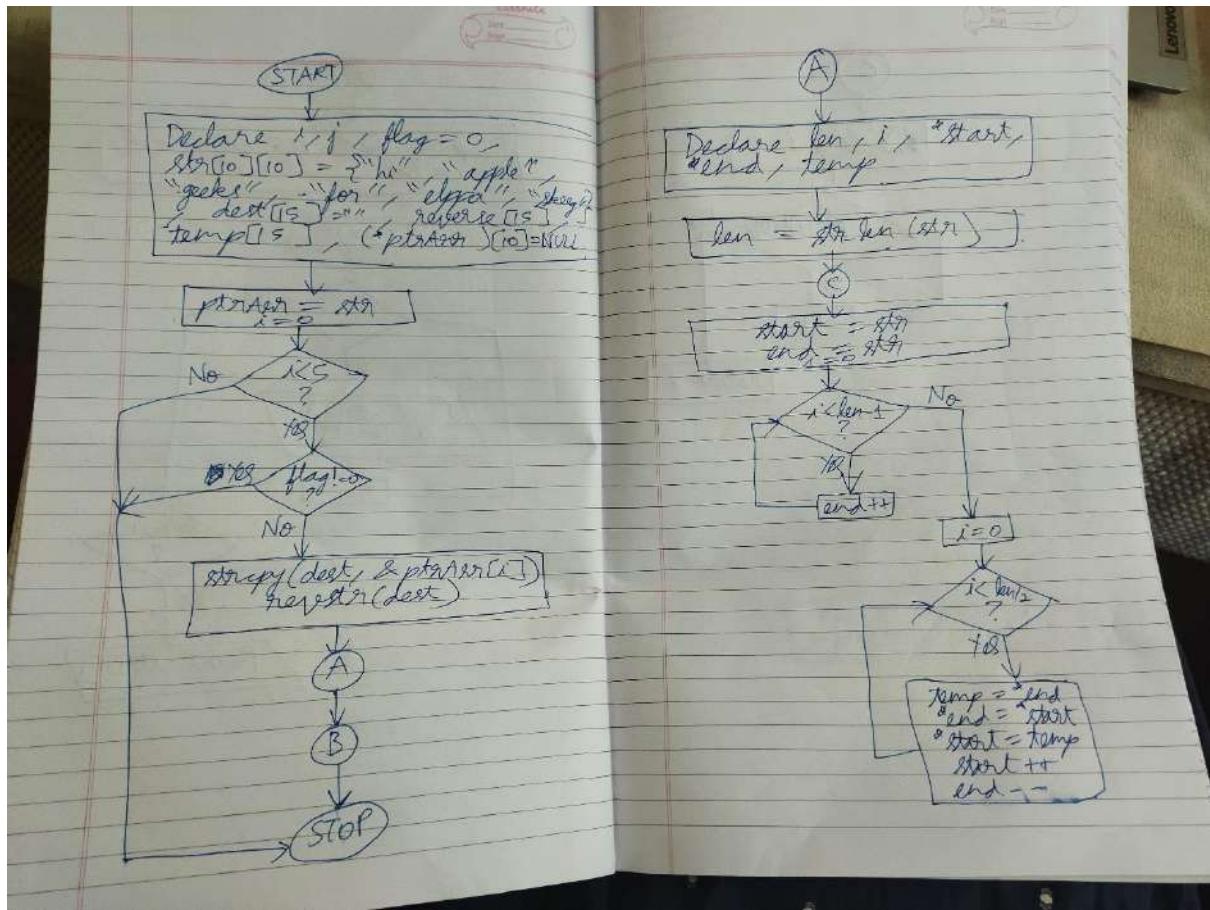
    return 0;
}
```

Inference- We are using pointers concept, and we are taking one string at a time and checking if the rest of the strings in

the array are same as it's reverse. When reverse is found, it's considered as first reverse string and the program stops.

CLASSEmate
Date _____
Page _____

1. START
2. Declare $i, j, flag \leftarrow 0$
 $str[10][10] \leftarrow \{hi\}$, "apple", "geeks"
"for", "elpa", "skeeg", "geeks", dest[10]
reverse[5], temp[5]
 $(^ptrArr)[10] \leftarrow \text{NULL}$
3. $ptrArr \leftarrow str$
4. for $i \leftarrow 0$ to 5 do
5. If $flag \neq 0$
6. break
7. strcpy(dest, &ptrArr[i])
8. reverse(dest)
9. for $j \leftarrow 0$ to 5 do
10. If $strcmp(dest, &ptrArr[j]) \neq 0$
11. $flag \leftarrow 1$
12. strcpy(temp, ptrArr[j])
13. reverse(temp)
14. strcpy(reverse, temp)
15. Print reverse, &ptrArr[i+1]
16. break
17. end for $j \leftarrow 5$
18. end for $i \leftarrow 5$
19. STOP



4. Create a new string by alternately combining the characters of two halves of the string in reverse.

Input : s = carbohydrates

Output : hsoebtraarcdy

Input : s = sunshine

Output : sennuish

Create New String by Alternatively Combining characters of 2 halves of the string in reverse

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

void print_reverse(char str[], int half_mark, int len) {
    int i, j, n=half_mark, m=len;
    for(i=n-1; i>=0; i--) {
        printf("%c", str[i]);
        printf("%c", str[m-1]);
        m--;
    }
    if(len%2 != 0)
        printf("%c", str[n]);
}

int main() {
    int half_mark, len;
    char str[100];
    printf("\nEnter string: ");
    scanf("%s", str);
    len = strlen(str);
    if(len%2 == 0)
        half_mark = len/2;
    else
        half_mark = (len-1)/2;
```

print reverse (str , half-mark , br)

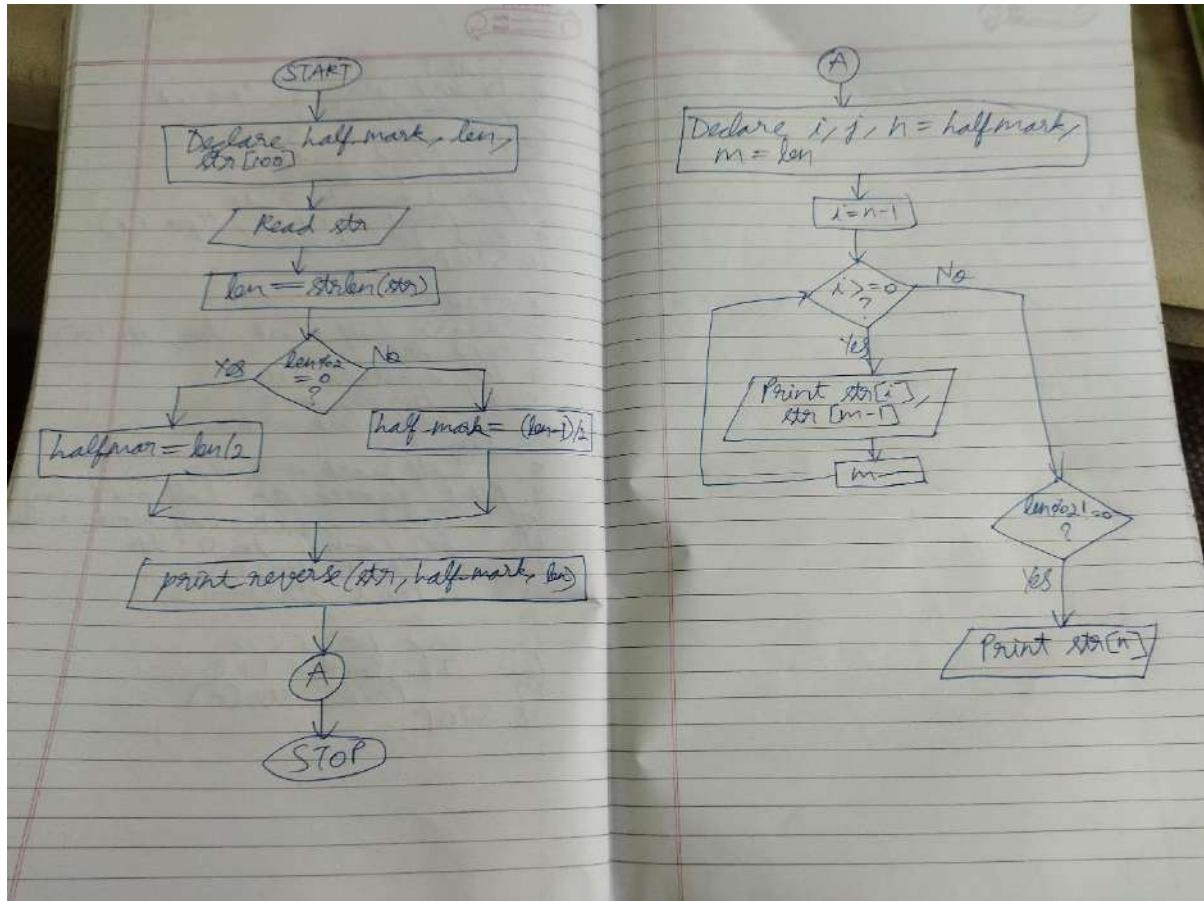
return 0;

}

Output - Enter string: sunshine
ennuish

Inference- We are alternatively combining both halves of the string to get the required string.

1. START
 2. Declare half-mark, len, str [100]
 3. Read str
 4. len \leftarrow strlen (str)
 5. If $len \% 2 = 0$
 6. half-mark \leftarrow len / 2
 7. Else
 8. half-mark \leftarrow (len - 1) / 2
 9. Print reverse (str, half-mark, len)
 10. Declare i, j, n = half-mark, m = len
 11. for i \leftarrow n - 1 to 0 do
 12. Print str[i]
 13. Print str[m - 1]
 14. m --
 15. end for i < 0
 16. If $len \% 2 != 0$
 17. Print str[n]
 18. STOP



LAB 15

Aim- Learn to use concept of structures.

1.Intervals: A range of consecutive integers is denoted by a starting and ending value, where the starting value is less than or equal to the ending value. For example, the range [18, 22] includes the integers 18, 19, 20, 21 and 22. Consider the task of writing a function that is given two ranges that determines whether or not the two ranges intersect. For example, [18,22] and [13, 19] do intersect, but [33, 88] and [90, 2000] do not. Complete the function below so that

it returns 1 if the two ranges specified intersect, and 0 if they do not. The first range will be [start1, end1] and the second range will be [start2, end2], where start1, end1, start2 and end2 are the four formal parameters to the function. Make sure to fill in the code according to the comments given.

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

```
struct range{  
    int start;  
    int end;  
};
```

```
int overlap(struct range interval1, struct range interval2)
{
    if(interval2.end < interval1.start)
        return 0;

    else if(interval1.end < interval2.start)
        return 0;

    else
        return 1;
}

int main()
{
    struct range inter1, inter2;

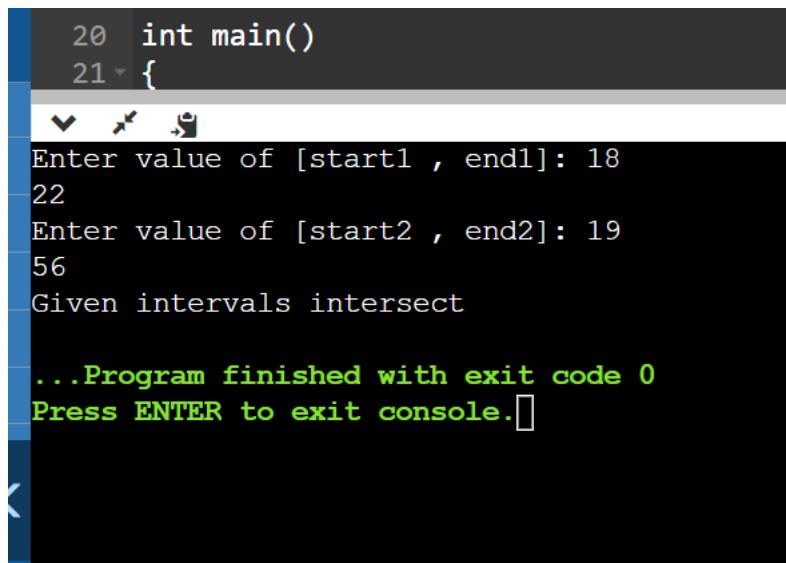
    printf("Enter value of [start1 , end1]: ");
    scanf("%d %d", &inter1.start, &inter1.end);

    printf("Enter value of [start2 , end2]: ");
    scanf("%d %d", &inter2.start, &inter2.end);
```

```
if(overlap(inter1, inter2))
printf("Given intervals intersect");

else
printf("Given intervals do not intersect");

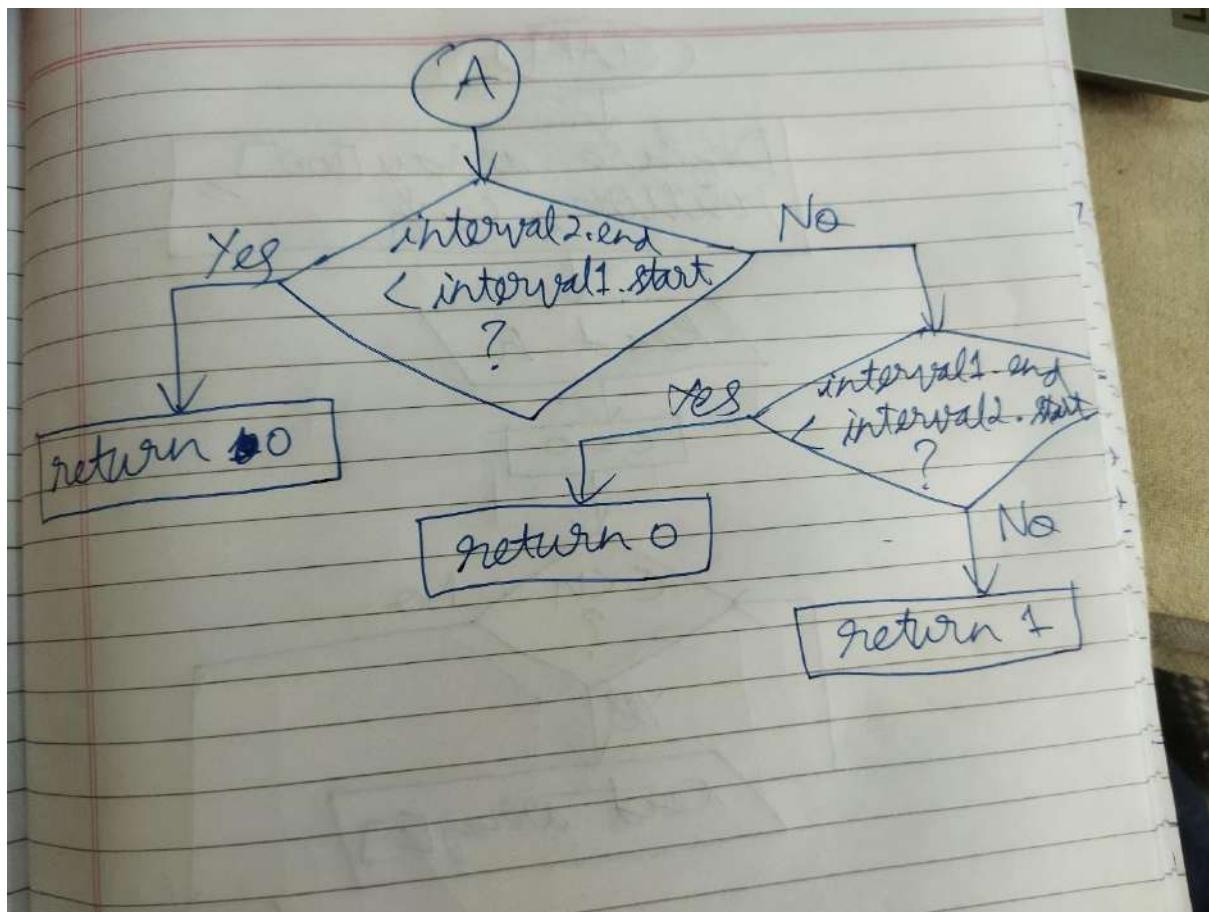
return 0;
}
```

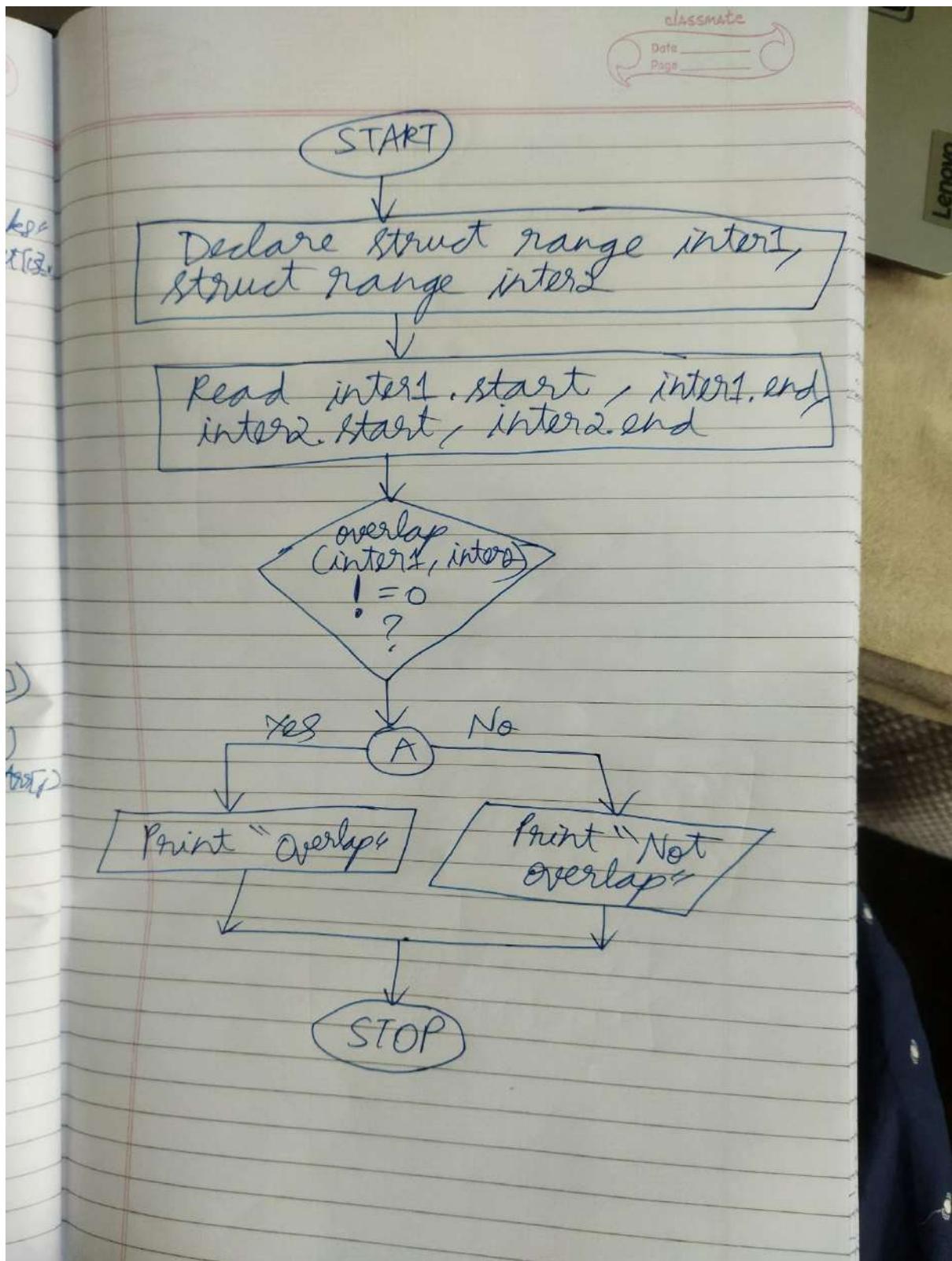


```
20 int main()
21 {
22
23     Enter value of [start1 , end1]: 18
24     Enter value of [start2 , end2]: 19
25     56
26     Given intervals intersect
27
28     ...Program finished with exit code 0
29     Press ENTER to exit console.[]
```

Inference- We are checking for non-overlapping condition, and based on that the program tells if the inputted range overlaps or not.

1. START
2. Declare ~~struct~~ range inter1, ~~struct~~ range inter2
3. Read inter1.start, inter1.end, inter2.start, inter2.end
4. If overlap(inter1, inter2) != 0
5. Print "Overlap"
6. Else
7. Print "No overlap"
8. STOP





2. Develop a C program using array of structures to store information

for student management system such as student roll no, student

name, Department, Course, CGPA and Year of joining. Perform the following using functions

- i. Search and display details of a particular student based on given roll no.
- ii. Display top 5 students based on CGPA obtained.
- iii. List out student names for the given department.

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

```
#include <string.h>
```

```
struct student{  
    int roll;  
    char name[50];  
    char department[25];  
    char course[25];  
    float cgpa;  
    int year_joined;  
}stud[100],t;
```

```
void display_details(int stu_num, struct student stud[])//if i don't  
pass stud[] inside,  
//it's not even reading values
```

```
{
```

```
int num,i,j, flag=0;
```

Samuela Abigail Mathew
71762108039

```
printf("\nEnter roll number of student: ");
scanf("%d",&num);

for(i=0;i<stu_num;i++)
{
if(stud[i].roll==num)
{
flag=1;
printf("\nThe details of the student are:");
printf("\nName: ");//i can't use %[^\n]s here
puts(stud[i].name);
printf("Roll number: %d",stud[i].roll);
printf("\nDepartment: ");
puts(stud[i].department);
printf("Course: ");
puts(stud[i].course);//puts() automatically implements \n at
end of string
printf("CGPA: %f",stud[i].cgpa);
printf("\nYear joined: %d",stud[i].year_joined);
}
}

}
```

if(!flag)

Samuela Abigail Mathew
71762108039

```
{  
printf("\nStudent not found!");  
}  
}  
  
void display_top(int stu_num, struct student stud[])  
{  
int i,j;  
  
for(i=0;i<stu_num;i++)  
{  
for(j=0;j<stu_num-1;j++)  
{  
if(stud[j].cgpa<stud[j+1].cgpa)  
{  
t=stud[j];  
stud[j]=stud[j+1];  
stud[j+1]=t;  
}  
}  
}  
  
printf("\nTop 5 students are:\n");  
for(i=0;i<5;i++)
```

Samuela Abigail Mathew
71762108039

```
{  
printf("Rank: %d CGPA: %f Name: ",i+1, stud[i].cgpa);  
puts(stud[i].name);  
}  
}
```

```
void list_dep(int stu_num, struct student stud[])
```

```
{  
int i, flag;  
char depname[25];
```

```
printf("\nEnter department name: ");
```

```
getchar();  
scanf("%[^\\n]s",depname);
```

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

```
{  
if(strcmp(depname,stud[i].department)==0)  
{  
flag=1;  
puts(stud[i].name);  
}  
}
```

```
if(!flag)  
printf("\nStudents not found in this department!");  
  
}
```

```
int main()  
{  
int option,n,i,j;  
struct student stud[100];
```

```
printf("\nEnter number of students whose details you want to  
enter: ");  
scanf("%d",&n);
```

```
for(i=0;i<n;i++)  
{  
printf("\nEnter details of student %d: ",i+1);  
printf("\nEnter name: ");  
getchar(); //if not used means scanf won't work for strings  
involving space characters inside loop  
scanf("%[^\\n]s",stud[i].name);  
printf("\nEnter roll number: ");  
scanf("%d",&stud[i].roll);
```

```
printf("\nEnter name of department: ");
getchar();
scanf("%[^\\n]s",stud[i].department);
```

```
printf("\nEnter name of course: ");
getchar();
scanf("%[^\\n]s",stud[i].course);
```

```
printf("\nEnter CGPA obtained: ");
scanf("%f",&stud[i].cgpa);
```

```
printf("\nEnter year of joining: ");
scanf("%d",&stud[i].year_joined);
```

```
}
```

```
printf("\nChoose an option:");
printf("\n1.Display details of a particular student.");
printf("\n2.Display top 5 students based on CGPA obtained.");
printf("\n3.List out student names for a particular department.");
printf("\n\nYour option is:");
scanf("%d",&option);
```

```
switch(option){
```

Samuela Abigail Mathew
71762108039

```
case 1:  
    display_details(n, stud);  
    break;  
  
case 2:  
    display_top(n,stud);  
    break;  
  
case 3:  
    list_dep(n, stud);  
    break;  
  
default:  
    printf("\nChoose a correct option!");  
    break;  
}  
  
return 0;  
}
```

```
Enter year of joining: 2021

Enter details of student 3:
Enter name: Subi K

Enter roll number: 29

Enter name of department: CSE

Enter name of course: Python

Enter CGPA obtained: 9

Enter year of joining: 2021

Choose an option:
1.Display details of a particular student.
2.Display top 5 students based on CGPA obtained.
3.List out student names for a particular department.

Your option is: 3

Enter department name: AI & DS
Samuela Abigail
Merlin Might

...Program finished with exit code 0
Press ENTER to exit console.[]
```

```
Enter roll number: 45
Enter name of department: CSE
Enter name of course: Python
Enter CGPA obtained: 9.6
Enter year of joining: 2021
Choose an option:
1.Display details of a particular student.
2.Display top 5 students based on CGPA obtained.
3.List out student names for a particular department.

Your option is: 1
Enter roll number of student: 39
The details of the student are:
Name: Sam Abi
Roll number: 39
Department: AI & DS
Course: C Programming
CGPA: 9.800000
Year joined: 2021
...Program finished with exit code 0
Press ENTER to exit console.[]
```

The screenshot shows a C IDE interface with the following components:

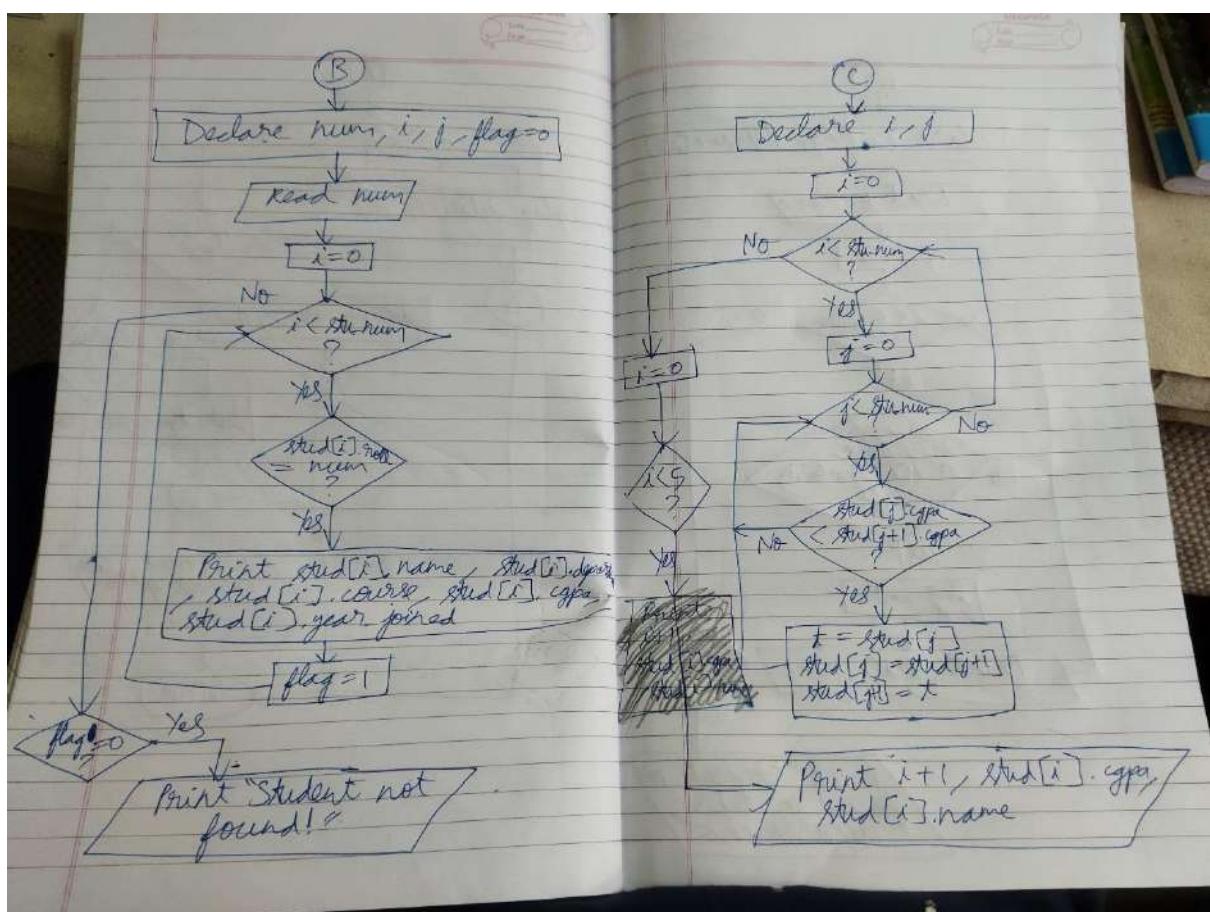
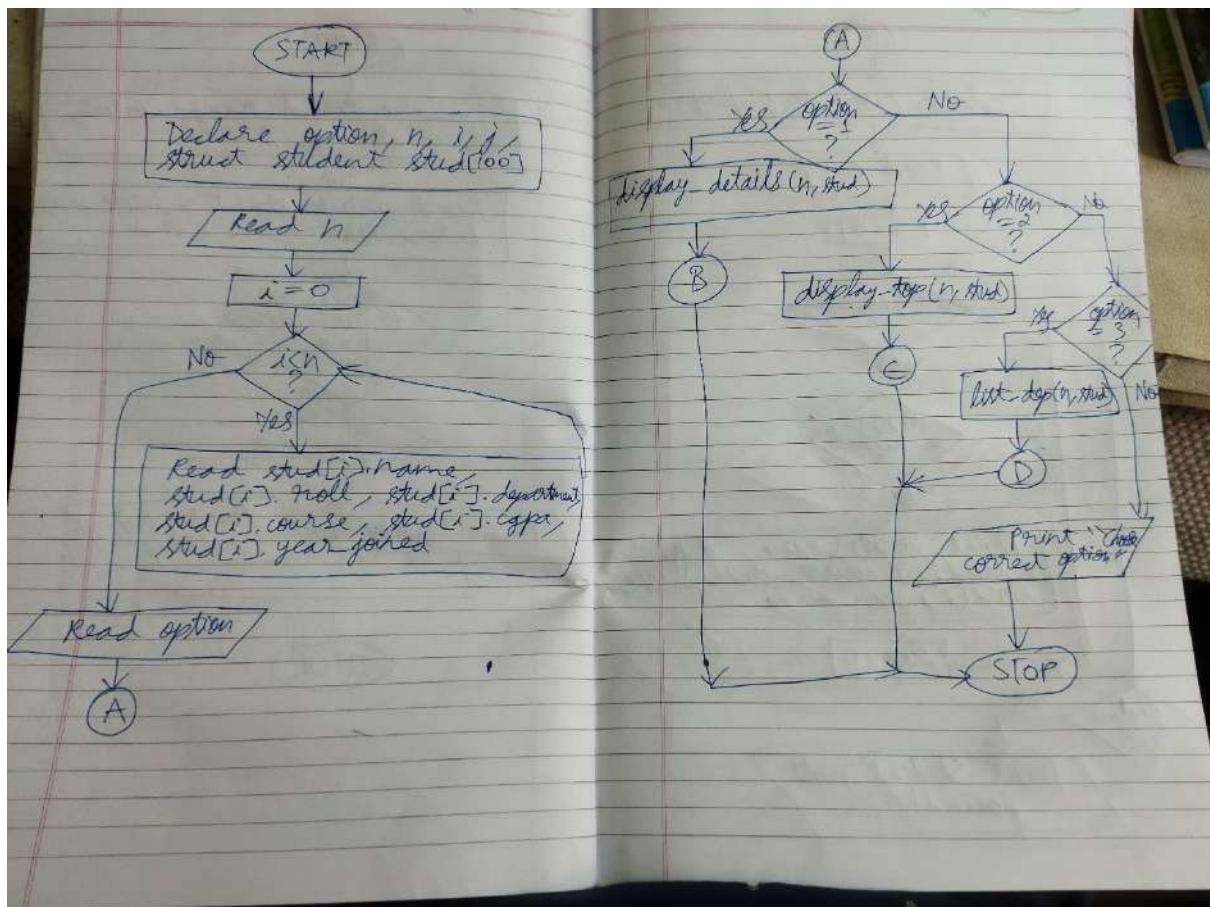
- Toolbar:** Includes icons for file operations (New, Open, Save), Run, Debug, Stop, Share, and Save.
- Code Editor:** The file `main.c` is open. The code contains a switch statement with three cases: 1, 2, and 3. Case 1 calls `display_details`. The code is as follows:

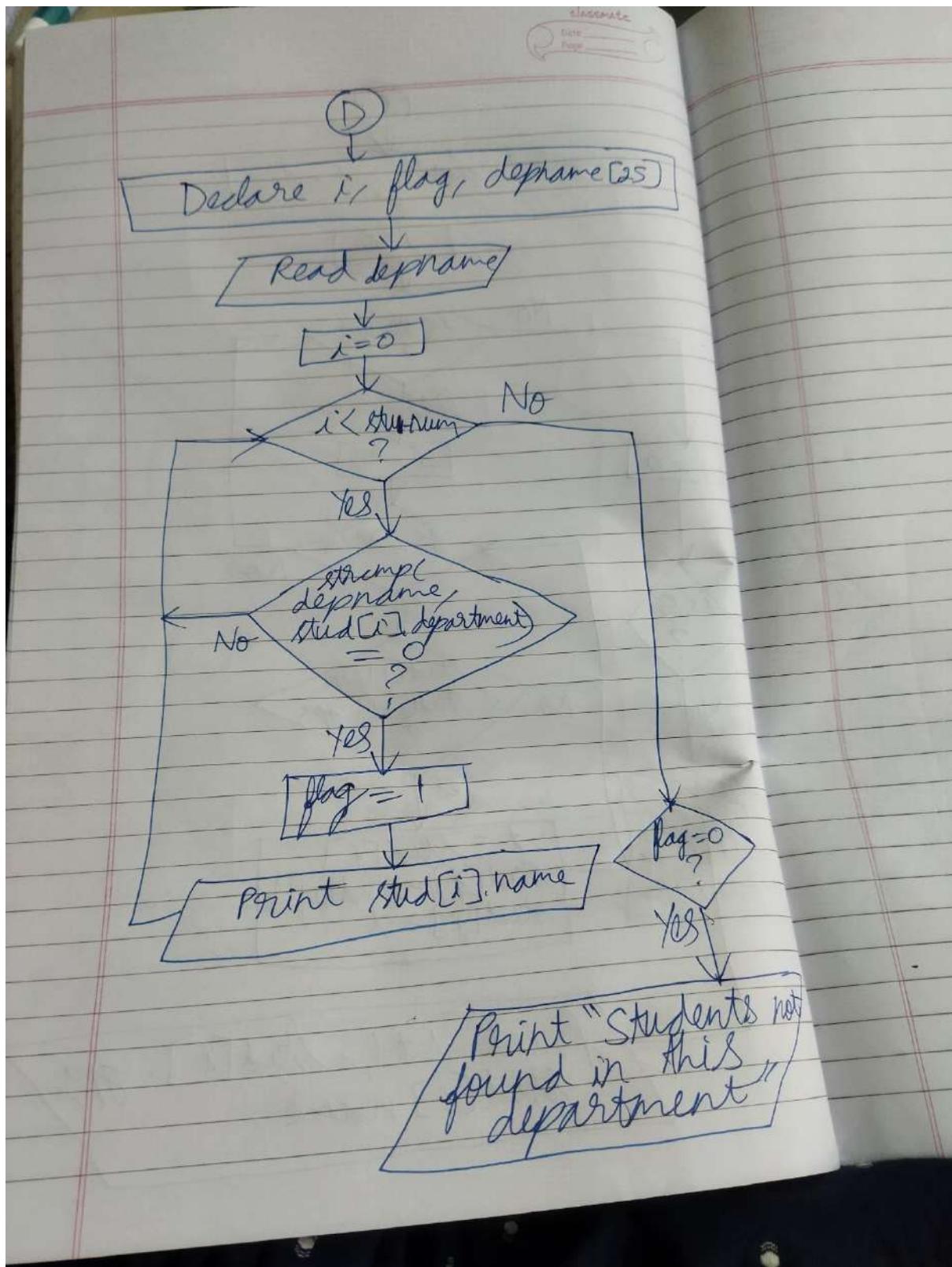
```
switch(option){  
    case 1:  
        display_details(n, stud);  
    break;  
}
```

- Output Console:** Displays the following user interactions and program output:
 - User enters "efw" as the course name.
 - User enters "7.98" as the CGPA.
 - User enters "32" as the year of joining.
 - User chooses option 1 to display student details.
 - Program outputs the top 5 students based on CGPA, listing them by rank (1 to 5) with their names and CGPAs.
- Status Bar:** Shows the message "...Program finished with exit code 0" and "Press ENTER to exit console."

Inference- structures and functions are used here.

1. START
2. Declare option
struct student {
 int i; //stud[100]
3. Read n
4. for i = 0 to n do
5. read stud[i].name
 stud[i].roll, stud[i].Department,
 stud[i].course, stud[i].cgpa
 stud[i].year_joined
6. end for i < n
7. Read option
8. switch(option){
9. case 1:
10. display_details(n, stud)
11. break
12. case 2:
13. display_top(n, stud)
14. break
15. case 3:
16. list_dep(n, stud)
17. break
18. default:
19. Print "choose correct option"
20. break
21. STOP





3. Create a structure called travel estimator, that holds the member variables like source, destination, fare, type of vehicle (Micro, Mini,

Samuela Abigail Mathew
71762108039

Sedan etc...) and the number of travellers using that vehicle. Write a menu driven program that performs the following operations.

- i) Calculate the total fare from the source to the final destination.
- ii) Calculate the average time taken for the entire journey.
- iii) List all the four places in some consecutive order by means of quickest time to reach from the source.

Use the information given in the below table to compute the above given options. The price per/km from the source is Rs.15. The trip starts from Aurangabad and the distance to each of the destinations from the source and the duration is listed below:

Destination	Distance	Trip Duration	Visiting Time
Mini Taj	7 km	20 mins	30 mins
Ghrishneshwar Temple	18 km	60 mins	20 mins
Daulatabad Fort	12 km	40 mins	90 mins
Ellora Caves	25 km	90 mins	180mins

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

```
#include <string.h>
```

```
struct travel_estimator{
```

```
int fare;
int Time;
char vehicle[25];
int travellers;
}trip;

int main()
{
int option;

printf("Choose your destination: ");
printf("\nDestination\tDistance\tTrip duration\tVisiting time");
printf("\n1.Mini Taj\t7km\t20min\t30min");
printf("\n2.Daulatabad Fort\t12km\t40min\t90min");
printf("\n3.Ghrishneshwar Temple\t18km\t60min\t20min");
printf("\n4.Ellora Caves\t25km\t90min\t180min");

printf("\n\nYour choice is: ");
scanf("%d",&option);

printf("\nEnter number of travellers: ");
scanf("%d",&trip.travellers);
printf("\nEnter name of vehicle you want to use from given
options: ");
```

```
printf("\nMicro\tMini\tSedan\tBus");
scanf("%s",trip.vehicle);
```

```
switch(option){
```

```
case 1:
```

```
    trip.fare=15*trip.travellers*7;
    trip.Time=20+30+20;
```

```
    break;
```

```
case 2:
```

```
    trip.fare=15*trip.travellers*12;
    trip.Time=40+90+40;
    break;
```

```
case 3:
```

```
    trip.fare=15*trip.travellers*18;
    trip.Time=60+20+60;
    break;
```

```
case 4:
```

```
    trip.fare=15*trip.travellers*25;
    trip.Time=90+180+90;
    break;
```

default:

```
printf("\nChoose correct option!");
break;
}
```

```
printf("\nYour total fare from Aurangabad to selected destination
```

```
is Rs%d",trip.fare);
```

```
printf("\nYour vehicle is %s and the number of travellers is
%d",trip.vehicle,trip.travellers);
```

```
printf("\nAverage time taken for entire journey (to and back) is %d
minutes",trip.Time);
```

```
return 0;
```

```
}
```

The screenshot shows a C IDE interface with a terminal window displaying the output of a program named 'main.c'. The terminal window has a title bar labeled 'input'.

```
Choose your destination:  
Destination Distance Trip duration Visting time  
1.Mini Taj 7km 20min 30min  
2.Daulatabad Fort 12km 40min 90min  
3.Ghrishneshwar Temple 18km 60min 20min  
4.Ellora Caves 25km 90min 180min  
  
Your choice is: 4  
  
Enter number of travellers: 5  
  
Enter name of vehicle you want to use from given options:  
Micro Mini Sedan Bus  
Sedan  
  
Your total fare from Aurangabad to selected destination is Rs1875  
Your vehicle is Sedan and the number of travellers is 5  
Average time taken for entire journey (to and back) is 360 minutes  
  
...Program finished with exit code 0  
Press ENTER to exit console.[]
```

Inference- structures and functions are used here.

1. START

2. Declare option

3. Read option, trip.travellers, trip.vehicle

4. Switch (option)

5. case 1:

6. trip.fare = 15 * trip.travellers * 7

7. trip.Time = 20 + 30 + 20

8. break

9. case 2:

10. trip.fare = 15 * trip.travellers * 12

11. trip.Time = 40 + 90 + 40

12. break

13. case 3:

14. trip.fare = 15 * trip.travellers * 18

15. trip.Time = 60 + 20 + 60

16. break

17. case 4:

18. trip.fare = 15 * trip.travellers * 25

19. trip.Time = 90 + 180 + 90

20. break

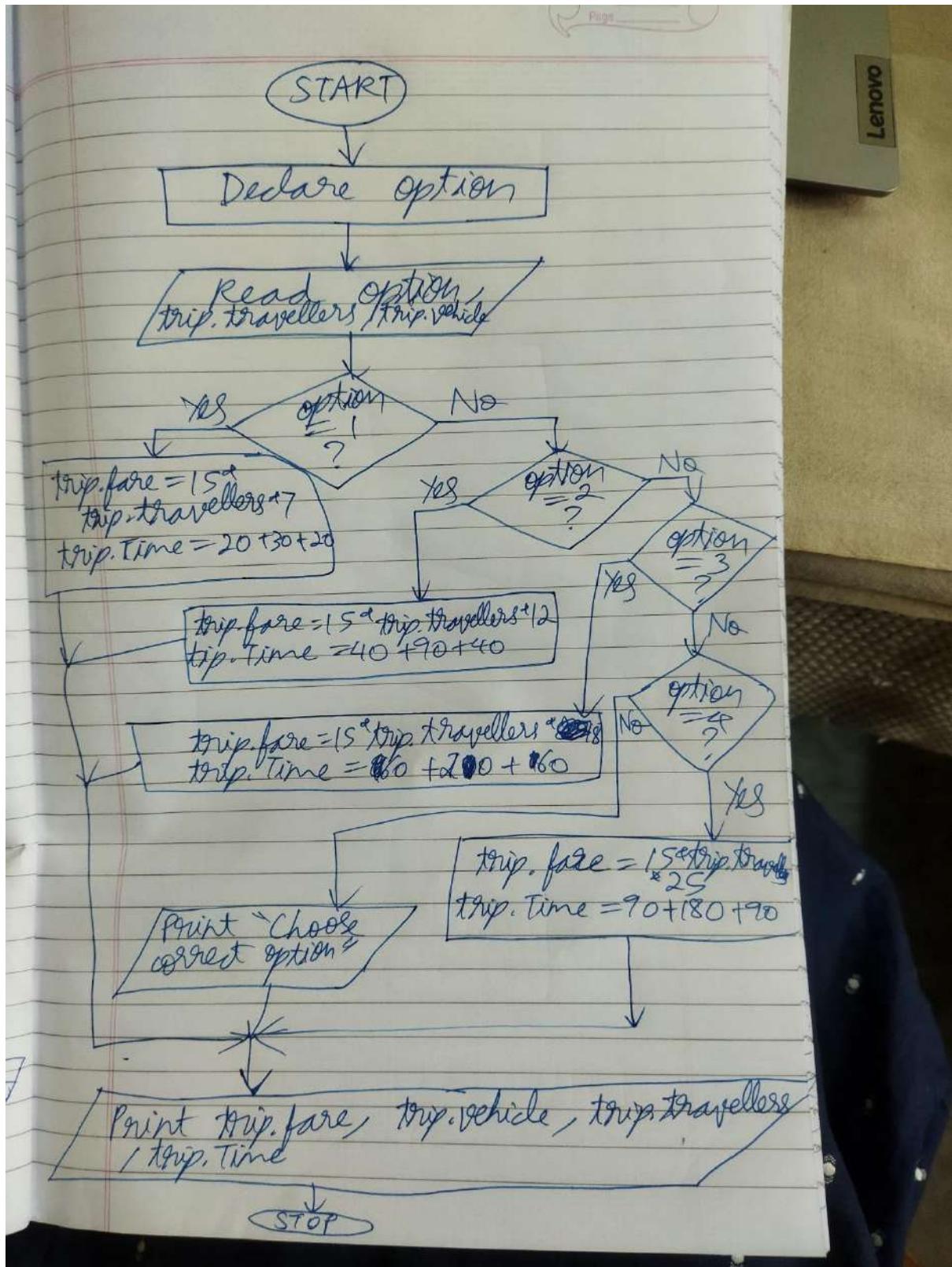
21. default:

22. Print "choose correct option"

23. break

24. Print trip.fare, trip.vehicle,
trip.travellers, trip.Time

25. STOP



LAB 16

Aim- Learn to use concept of files.

1) Reversal of words: Assume that you are provided with an input text file (ip.txt) which contains a few lines from a poem. The task is to read the file and reverse all the words present in it. The output must be stored in an output file, say, op.txt.

Example:

Input:

Miles to go before I sleep

Miles to go before I sleep

Output:

seliM ot og erofeb I peels

seliM ot og erofeb I peels

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

```
#include <string.h>
```

```
int main()
```

```
{
```

```
char buf[100], str[100], temp;
```

```
char rev[100][200], ch;  
int line = 0, i = 0, j=0, len;  
FILE *fptr;  
  
FILE* fp = fopen("ip.txt", "r");  
  
if (fp == NULL) {  
    printf("Unable to open file\n");  
    return -1;  
}  
  
while ((ch=fgetc(fp))!=EOF) {  
    if(ch=='\n')  
        line++; //counting lines  
}  
  
// Move the pointer back to 0th index  
rewind(fp);  
  
printf("\nContent of ip.txt: \n");  
for (i=0;i<line;i++) {  
    fgets(buf, sizeof(buf), fp);
```

Samuela Abigail Mathew
71762108039

```
strcpy(rev[i],buf);
printf("%s", buf);
}
```

```
fclose(fp);
```

```
fptr=fopen("op.txt","w");
```

```
if (fptr == NULL) {
printf("\nUnable to open file");
return -1;
}
```

```
printf("\n\nContent of op.txt: ");
while(j<line){
```

```
strcpy(str,rev[j]);
len=strlen(str);
for (i = 0; i < len/2; i++)
{
// temp variable use to temporary hold the string
temp = str[i];
```

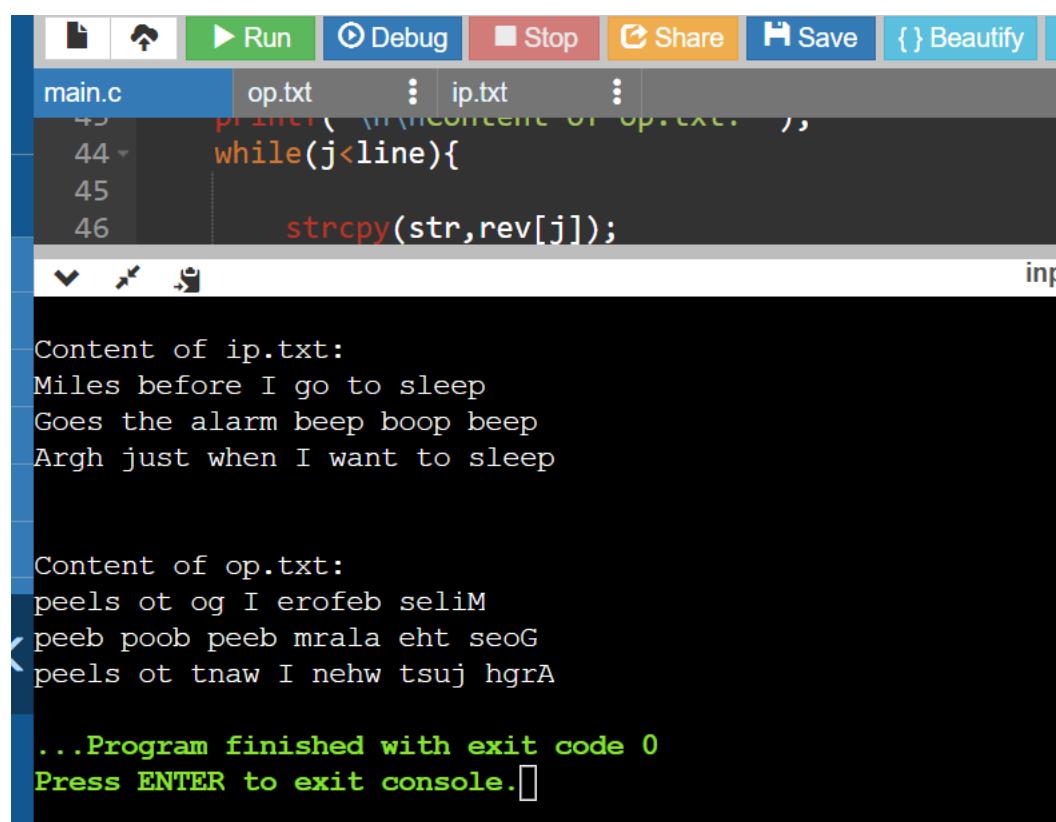
Samuela Abigail Mathew
71762108039

```
str[i] = str[len - i - 1];  
str[len - i - 1] = temp;  
}  
  
fprintf(fptr,"%s",str);  
printf("%s",str);  
j++;  
}
```

```
fclose(fptr);
```

```
return 0;
```

```
}
```

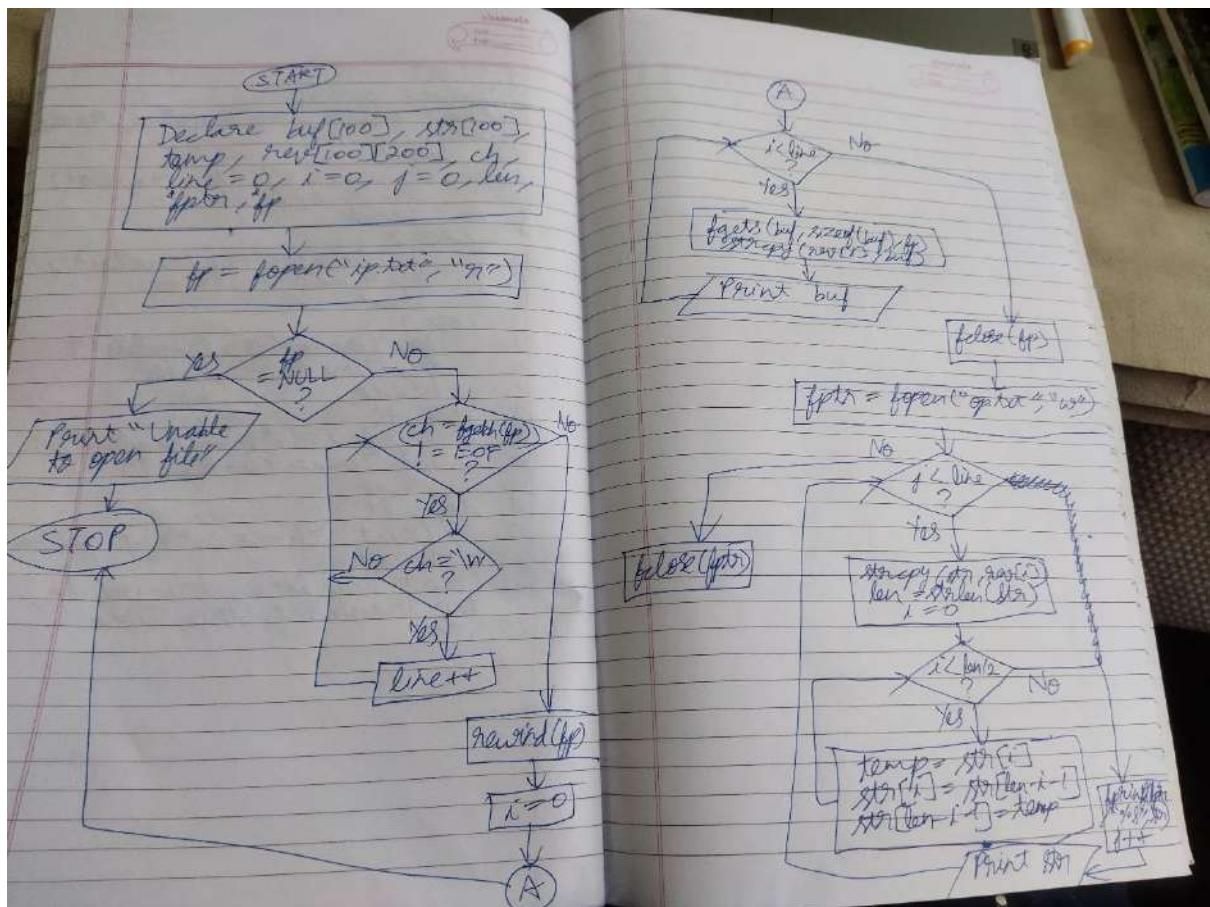


```
Content of ip.txt:  
Miles before I go to sleep  
Goes the alarm beep boop beep  
Argh just when I want to sleep  
  
Content of op.txt:  
peels ot og I erofeb seliM  
peeb poob peeb mrala eht seoG  
peels ot tnav I nehw tsuj hgrA  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

```
1 Miles before I go to sleep
2 Goes the alarm beep beep beep
3 Argh just when I want to sleep
4
```

```
1 peels ot og I erofeb seliM
2 peeb poob peeb mrala eht seoG
3 peels ot tnav I nehw tsuj hgrA
4
```

Inference- We are reversing position of each character in each sentence in the text file ip.txt and storing the resultant text in another text file op.txt .



2) Reversal of lines: Assume that you are provided with an input text file (ip.txt) which contains a few lines from a poem. The task is to read the file and reverse all the position of the words present in it. The output must be stored in an output file, say, op.txt.

Example:

Input:

Miles to go before I sleep

Miles to go before I sleep

Output:

sleep I before go to Miles

sleep I before go to Miles

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

void reverse(char str[], FILE *fptr)
{
    int i,j,len,startIndex, endIndex;

    len = strlen(str);
    endIndex = len - 1;

    for(i = len - 1; i >= 0; i--)
    {
        if(str[i] == ' ' || i == 0)
        {
            if(i == 0)
            {
                startIndex = 0;
            }
            else
            {

```

Samuela Abigail Mathew
71762108039

```
startIndex = i + 1;  
}  
  
for(j = startIndex; j <= endIndex; j++)  
{  
    fprintf(fptr,"%c",str[j]);  
    printf("%c", str[j]);  
}  
  
endIndex = i - 1;  
fprintf(fptr," ");  
printf(" ");  
}  
}  
  
}
```

```
int main()  
{  
char buf[100], str[100];  
char arr[100][100], ch;  
int line = 0, i = 0;  
FILE *fptr;
```

```
FILE* fp = fopen("ip.txt", "r");

if (fp == NULL) {
    printf("Unable to open file\n");
    return -1;
}

while ((ch=fgetc(fp))!=EOF) {
    if(ch=='\n')
        line++;//counting lines
}

// Move the pointer back to 0th index
rewind(fp);

printf("\nContent of ip.txt: \n");
for (i=0;i<line;i++) {
    fgets(buf, sizeof(buf), fp);
    strcpy(arr[i],buf);
    printf("%s", buf);
}
```

```
fclose(fp);

fptr=fopen("op.txt","w");

if (fptr == NULL) {
    printf("\nUnable to open file");
    return -1;
}

printf("\n\nContent of op.txt:\n ");
for(i=0;i<line;i++)
{
    strcpy(str,arr[i]);

    reverse(str, fptr);
    //fprintf(fptr, "\n");
    //printf("\n");
}

fclose(fptr);
```

Samuela Abigail Mathew
71762108039

```
return 0;
```

```
}
```

```
Run Debug Stop Share Help
main.c ip.txt op.txt
70     if ((fp1 = fopen(ip, "r")) == NULL) {
71         printf("\nUnable to open file")
72         return -1;
73     }

Content of ip.txt:
Hello Sam
I am Daniela Johanna
I like cookies

Content of op.txt:
Sam Hello
Johanna Daniela am I
cookies like I

...Program finished with exit code 0
Press ENTER to exit console.
```

```
Run Debug Stop Share
main.c ip.txt op.txt
1 Hello Sam
2 I am Daniela Johanna
3 I like cookies
4
```

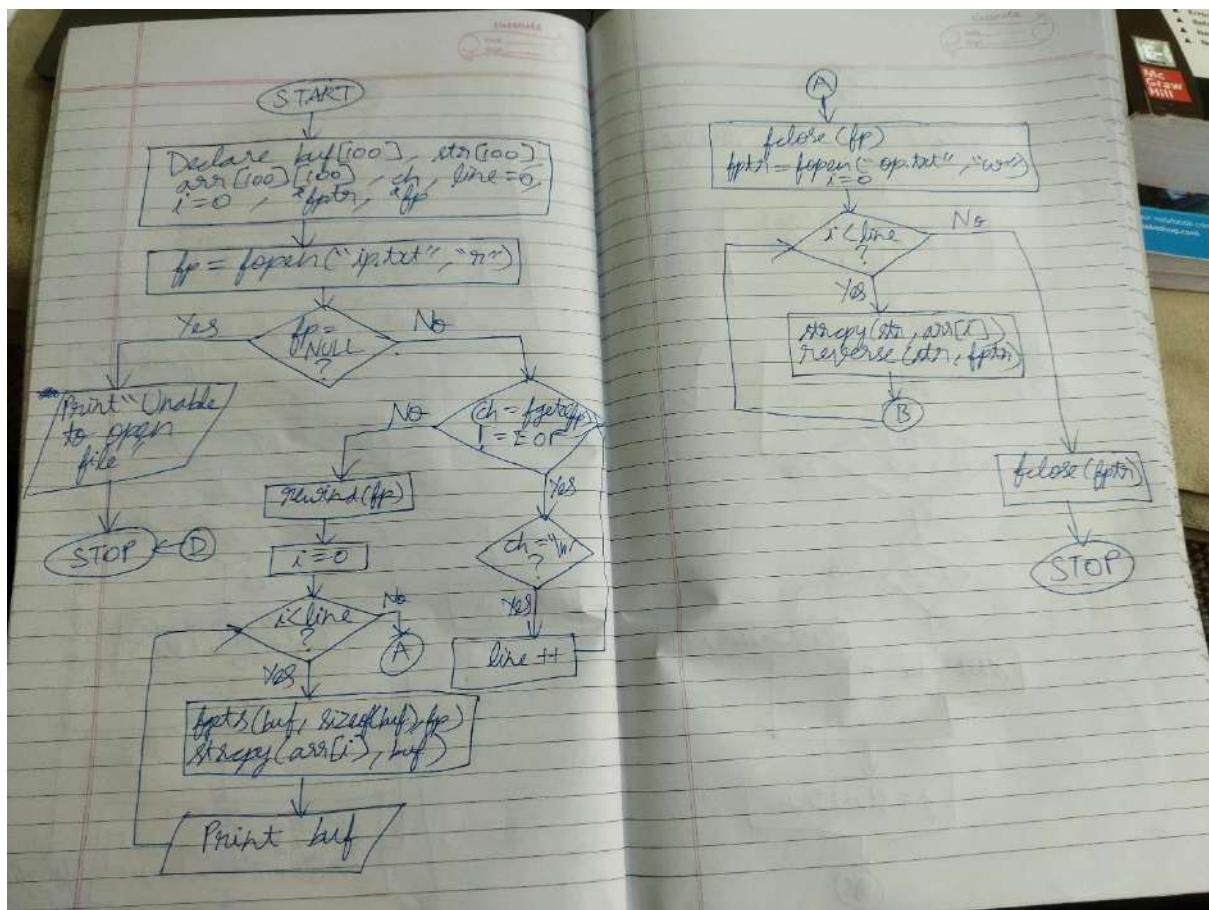
```

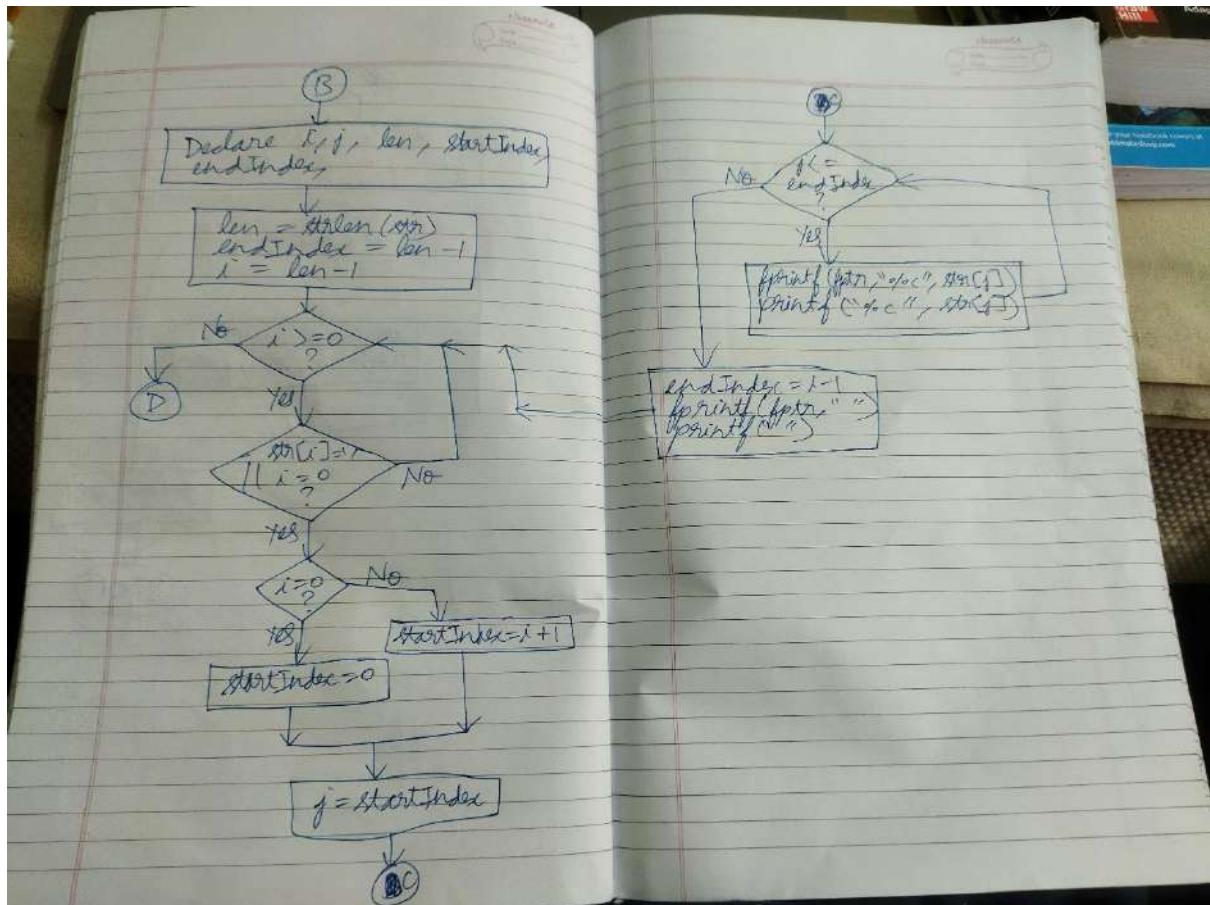
main.c          ip.txt          :: op.txt ::

1
2 Sam Hello
3 Johanna Daniela am I
4 cookies like I

```

Inference- We are reversing position of each word in each sentence in the text file ip.txt and storing the resultant text in another text file op.txt .





3. Archiving Record in a File System: Assume that there are 20 students in a class. Each student academic information is represented as an individual record. The attributes of the record are as follows:

Reg no – int

Name – char array

Sem – int

Cgpa – float

All the student information records have to be stored in a file

grad.bin. You are asked to insert a new record at the last, delete the first record, manipulate the kth ($1 \leq k \leq 20$) record in the grad.txt. Write a program for the same.

Note: Create a structure for the student record. The structure can be written and or fetched from the file (grad.bin). Use binary files reading and writing operations to deploy the tasks.

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

```
struct student{  
    int reg;  
    char name[50];  
    int sem;  
    float cgpa;  
}stud[21], temp;
```

```
void insert(struct student stud[]){
```

```
    int i;
```

```
FILE *fpt=fopen("grad.bin","rb");  
if(fpt==NULL){  
    printf("\nUnable to open file");
```

```
return;
```

```
}
```

```
FILE *fptr=fopen("temp.bin","wb");
```

```
if(fpt==NULL){
```

```
printf("\nUnable to open file");
```

```
return;
```

```
}
```

```
for(i=0;i<20;i++){
```

```
fread(&stud[i], sizeof(struct student), 1,fpt);
```

```
fwrite(&stud[i],sizeof(struct student),1,fptr);
```

```
}
```

```
printf("\nEnter student details: ");
```

```
printf("\nName: ");
```

```
getchar();
```

```
scanf("%[^\\n]s",stud[20].name);//we can't calculate no. of  
lines in
```

```
binary files
```

//so I'm assuming new student is 21st student since 20
students

are already in class

```
printf("\nRegistered number: ");  
scanf("%d",&stud[20].reg);
```

```
printf("\nSemester number: ");  
scanf("%d",&stud[20].sem);
```

```
printf("\nCGPA obtained: ");  
scanf("%f",&stud[20].cgpa);
```

```
fwrite(&stud[20],sizeof(struct student),1,fptr);
```

```
fclose(fpt);
```

```
fclose(fptr);
```

```
remove("grad.bin");  
rename("temp.bin","grad.bin");
```

```
fpt=fopen("grad.bin","rb");
```

```
printf("\nUpdated Student Archive:");
printf("\nName\t\tRegistered number\tSemester
number\tCGPA\n");
for(i=0;i<21;i++){
fread(&stud[i], sizeof(struct student), 1,fpt);

printf("%s\t\t%d\t\t%d\t\t%f\n",stud[i].name,stud[i].reg,stu
d[i].sem,
stud[i].cgpa );
//it's not printing first character of first name, online
compiler
problem ig
//in codeblocks it's working fine if not present in same
directory
}

fclose(fpt);

}

void delete(struct student stud[]){

```

```
int del, i;
```

```
FILE *fpt=fopen("grad.bin","rb");  
if(fpt==NULL){  
printf("\nUnable to open file");  
return;  
}
```

```
FILE *fptr=fopen("temp.bin","wb");  
if(fpt==NULL){  
printf("\nUnable to open file");  
return;  
}
```

```
printf("\nEnter registered number of student whose records  
need  
to be deleted: ");  
//since registered number is supposed to be unique, I'm  
considering it to identify student  
scanf("%d",&del);
```

```
for(i=0;i<21;i++){
```

```
fread(&stud[i], sizeof(struct student), 1,fpt);

if(del!=stud[i].reg)
fwrite(&stud[i],sizeof(struct student),1,fptr);
}

fclose(fptr);
fclose(fpt);

remove("grad.bin");
rename("temp.bin","grad.bin");

fpt=fopen("grad.bin","rb");

printf("\nUpdated Student Archive:");
printf("\nName\t\tRegistered number\tSemester
number\tCGPA\n");
for(i=0;i<20;i++){
fread(&stud[i], sizeof(struct student), 1,fpt);
```

```
printf("%s\t%d\t%d\t%f\n",stud[i].name,stud[i].reg,stu
d[i].sem,
stud[i].cgpa );
//it's not printing first character of first name, online
compiler
problem ig
//in codeblocks it's working fine if not in same directory
}

fclose(fpt);
```

```
void modify(struct student stud[]){
int mod, i;

FILE *fpt=fopen("grad.bin","rb");
if(fpt==NULL){
printf("\nUnable to open file");
return;
}
```

```
FILE *fptr=fopen("temp.bin","wb");
if(fpt==NULL){
printf("\nUnable to open file");
return;
}

printf("\nEnter registered number of student whose records
need
to be modified: ");
//since registered number is supposed to be unique, I'm
considering it to identify student
scanf("%d",&mod);

for(i=0;i<21;i++){
fread(&stud[i], sizeof(struct student), 1,fpt);

if(mod==stud[i].reg)

{
printf("\nEnter details: ");
printf("\nName: ");


```

```
getchar();
scanf("%[^\\n]s",stud[i].name);

printf("\nRegistered number: ");
scanf("%d",&stud[i].reg);

printf("\nSemester number: ");
scanf("%d",&stud[i].sem);

printf("\nCGPA Obtained: ");
scanf("%f",&stud[i].cgpa);

fwrite(&stud[i],sizeof(struct student),1,fptr);

}

else
{
fwrite(&stud[i],sizeof(struct student),1,fptr);
}
}
```

```
fclose(fptr);

fclose(fpt);

remove("grad.bin");
rename("temp.bin","grad.bin");

fpt=fopen("grad.bin","rb");

printf("\nUpdated Student Archive:");
printf("\nName\t\tRegistered number\tSemester
number\tCGPA\n");
for(i=0;i<21;i++){
fread(&stud[i], sizeof(struct student), 1,fpt);

printf("%s\t\t%d\t\t%d\t\t%f\n",stud[i].name,stud[i].reg,stu
d[i].sem,
stud[i].cgpa );
//it's not printing first character of first name, online
compiler
problem ig
//in codeblocks it's working fine if not in same directory
}
```

```
fclose(fpt);
```

```
}
```

```
int main()
```

```
{
```

```
int option, i;
```

```
char ch;
```

```
FILE *fptr;
```

```
fptr=fopen("grad.bin","wb");
```

```
if(fptr==NULL)
```

```
{
```

```
printf("\nCan't open file");
```

```
return -1;
```

```
}
```

```
printf("\nEnter details of 20 students: ");
```

```
//this part is not needed if grad.bin already has existing  
student  
records
```

```
for (i=0;i<20;i++) {  
    printf("\nEnter details of student %d:",i+1);  
    printf("\nName: ");  
    getchar();  
    scanf("%[^\\n]s",stud[i].name);  
  
    printf("\nRegistered number: ");  
  
    scanf("%d",&stud[i].reg);  
  
    printf("\nSemester number: ");  
    scanf("%d",&stud[i].sem);  
  
    printf("\nCGPA Obtained: ");  
    scanf("%f",&stud[i].cgpa);  
  
    fwrite(&stud[i],sizeof(struct student),1,fptr);  
}  
  
fclose(fptr);  
  
printf("\nChoose your option: ");
```

Samuela Abigail Mathew
71762108039

```
printf("\n1.Insert student record");
printf("\n2.Delete student record");
printf("\n3.Modify student record");
```

```
printf("\n\nYour choice is: ");
scanf("%d",&option);
```

```
switch(option){
```

```
case 1:
```

```
insert(stud);
```

```
break;
```

```
case 2:
```

```
delete(stud);
```

```
break;
```

```
case 3:
```

```
modify(stud);
```

```
break;
```

```
default:
```

```
printf("\nEnter correct option!");
```

```
break;
```

```
}
```

```
return 0;
```

```
}
```

```
CGPA obtained: 3.86

Updated Student Archive:
Name          Registered number      Semester number CGPA
Samuela Abigail Mathew      39           1             9.675800
Daniela Johanna        7            0             7.985000
Sheeba Rani          34           8             9.385000
Suresh Mathew         1            9             9.874500
Jeremy m             2            0             2.946700
Josh Ore              71           11            8.340000
Joshua Selman        67           15            5.856700
Jennifer LeClaire     32           18            9.670000
Robert Clancy         12           16            8.236000
Jenni N               37           1             9.000000
Ancy                 19           1             5.230000
Janet H               123          1             4.270000
Carol LH              54           6             7.580000
Jane sd               17           0             2.670000
Medea S               43           12            10.000000
Airis Riedel          19           1             8.900000
Raj GK                16           12            2.900000
TNT                  178          8             10.000000
ARA                  29           15            3.780000
jk rose               190          2             8.370000
Ambrose Aria          2355          5             3.860000

...Program finished with exit code 0
Press ENTER to exit console.
```

The screenshot shows a C IDE interface with a toolbar at the top containing icons for Run, Debug, Stop, Share, Save, Beautify, and a download arrow. The Language is set to C. Below the toolbar, there are tabs for main.c, grad.bin, temp.bin, and filename.bin. The main code editor window contains the following C code:

```
1 //amuela Abigail Mathew.....D.A.....Daniela Johanna.....
2
```

Below the code editor is a terminal window titled "input" showing the following command-line interaction:

```
jk rose      190      2      8.370000
Ambrose Aria    2355      5      3.860000

...Program finished with exit code 0
Press ENTER to exit console.
```

Inference- We are using binary file to store student records and perform given conditions.

