

C – Programming

>> System Specifications :

Processor > Ryzen 5 5600h (6 cores 12 threads) 3.30Ghz base speed

Memory > 8GB ram

Graphics > NVIDIA GeForce RTX 3050 (4gb Vram)

Storage > 512Gb internal SSD

Software : Visual Studio Code (Version 1.93)

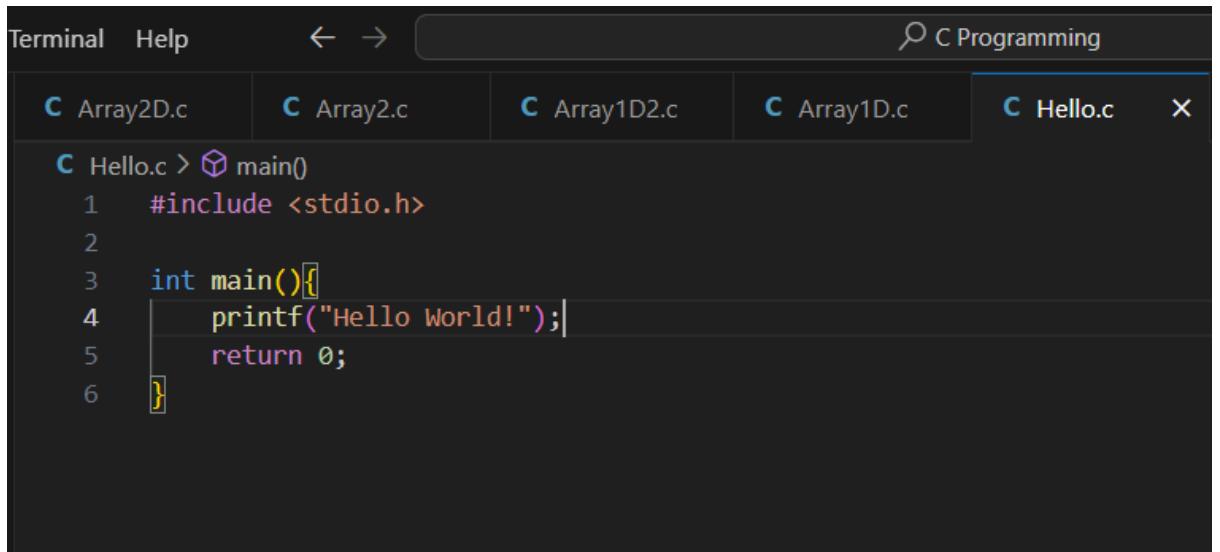
>> Name – Sankalp Pandey

>>Batch – 71

>>Sap_ID- 590015862

C - Programs :

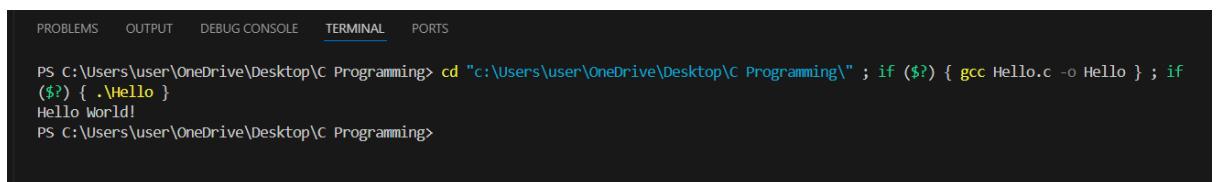
1) Print "Hello World!"



The screenshot shows a code editor interface with a dark theme. At the top, there's a navigation bar with 'Terminal', 'Help', and search icons. Below the bar, there are tabs for several C files: 'Array2D.c', 'Array2.c', 'Array1D2.c', 'Array1D.c', 'Hello.c', and a closed tab. The 'Hello.c' tab is currently active, displaying the following C code:

```
1 #include <stdio.h>
2
3 int main(){
4     printf("Hello World!");
5     return 0;
6 }
```

Output :



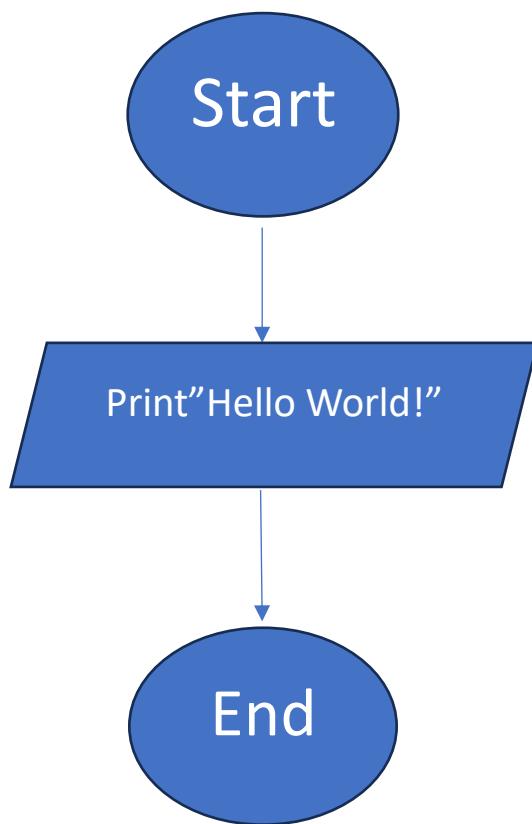
The screenshot shows a terminal window with a dark theme. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected), and 'PORTS'. The terminal window displays the following command-line session:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming" ; if ($?) { gcc Hello.c -o Hello } ; if
($?) { .\Hello }
Hello World!
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

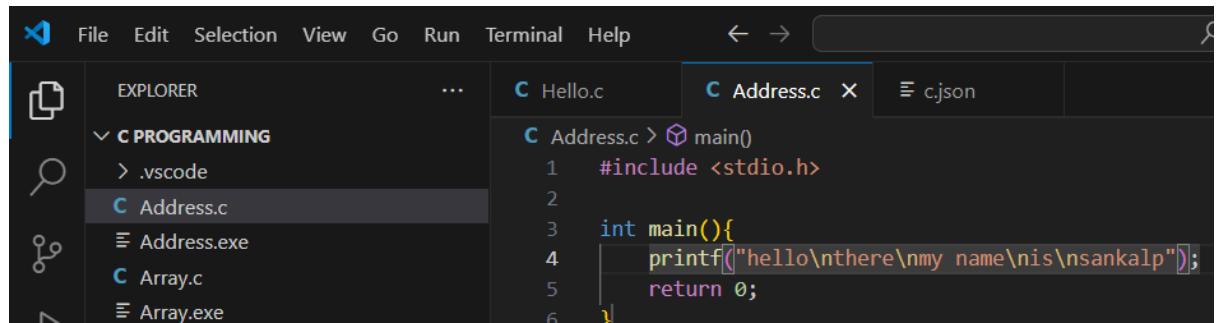
Algorithm :

- 1) Start
- 2) Include stdio.h (header)
- 3) Starting main () function
- 4) Printing the statement “Hello World!”
- 5) End

FLOW CHART :

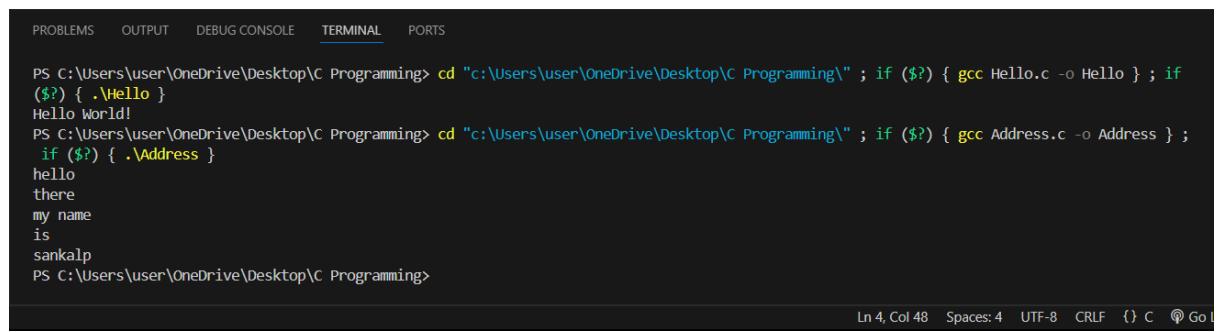


2) Print the address in multiple lines (new line)



```
#include <stdio.h>
int main(){
    printf("hello\nthere\nmy name\nis\nsankalp");
    return 0;
}
```

Output :

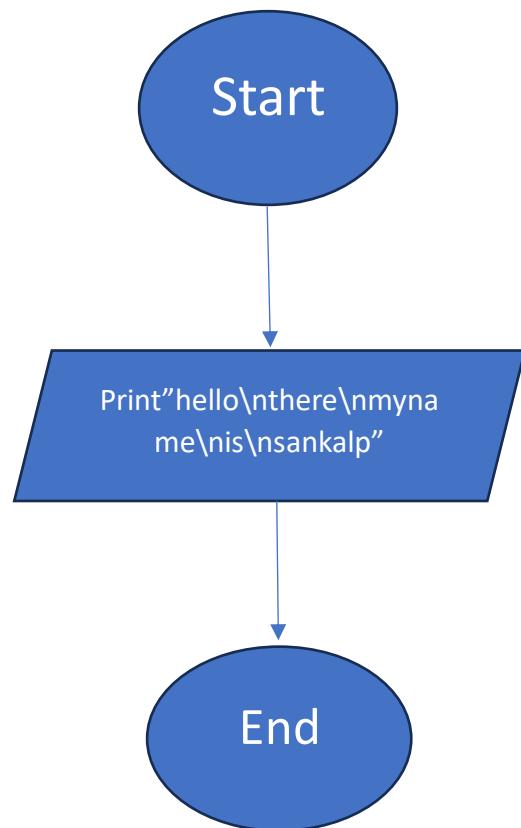


```
PS C:\Users\user\Desktop\C Programming> cd "c:\Users\user\Desktop\C Programming"
PS C:\Users\user\Desktop\C Programming> if ($?) { gcc Hello.c -o Hello }
PS C:\Users\user\Desktop\C Programming> if ($?) { gcc Address.c -o Address }
PS C:\Users\user\Desktop\C Programming> ./Address
Hello World!
hello
there
my name
is
sankalp
PS C:\Users\user\Desktop\C Programming>
```

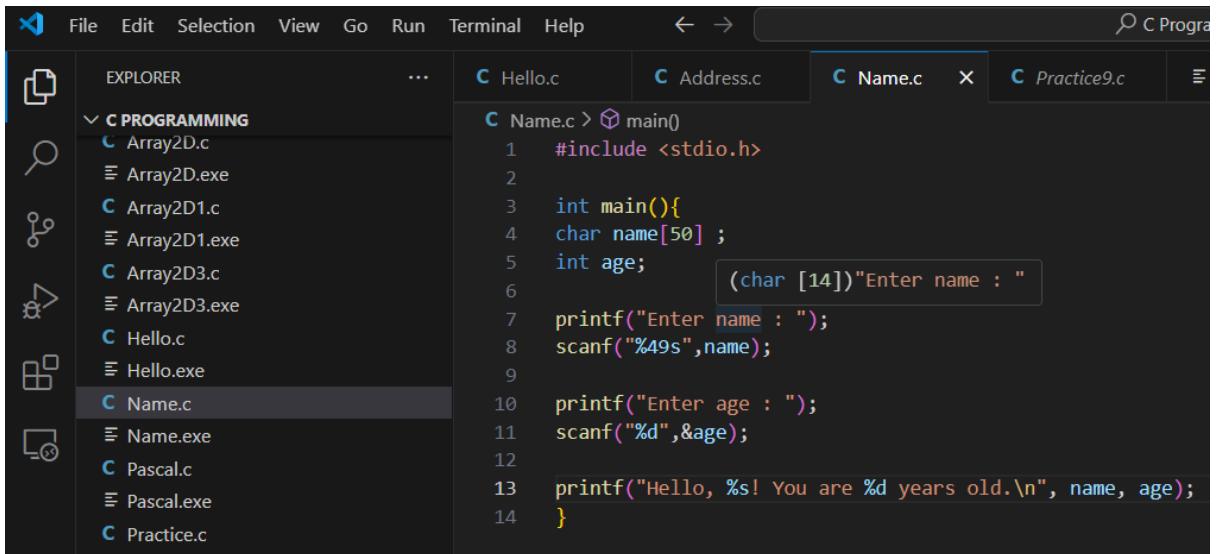
Algorithm :

- 1) Start
- 2) Include header file
- 3) Starting main() function
- 4) Enter address (use printf)
- 5) End

FLOW CHART :



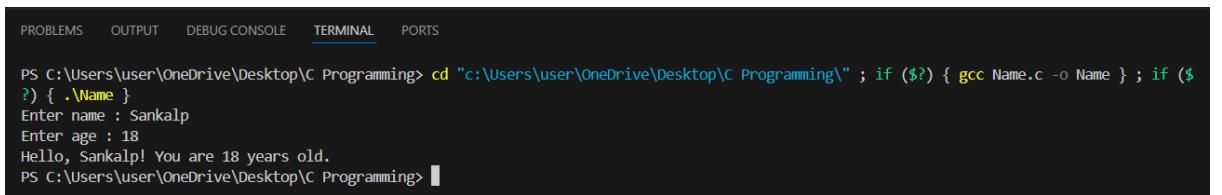
3) that prompts the user to enter their name and age.



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar showing files in a folder named 'C PROGRAMMING'. The 'Name.c' file is selected. The main area displays the following C code:

```
#include <stdio.h>
int main(){
    char name[50];
    int age;
    (char [14])"Enter name : "
    printf("Enter name : ");
    scanf("%49s",name);
    printf("Enter age : ");
    scanf("%d",&age);
    printf("Hello, %s! You are %d years old.\n", name, age);
}
```

Output :



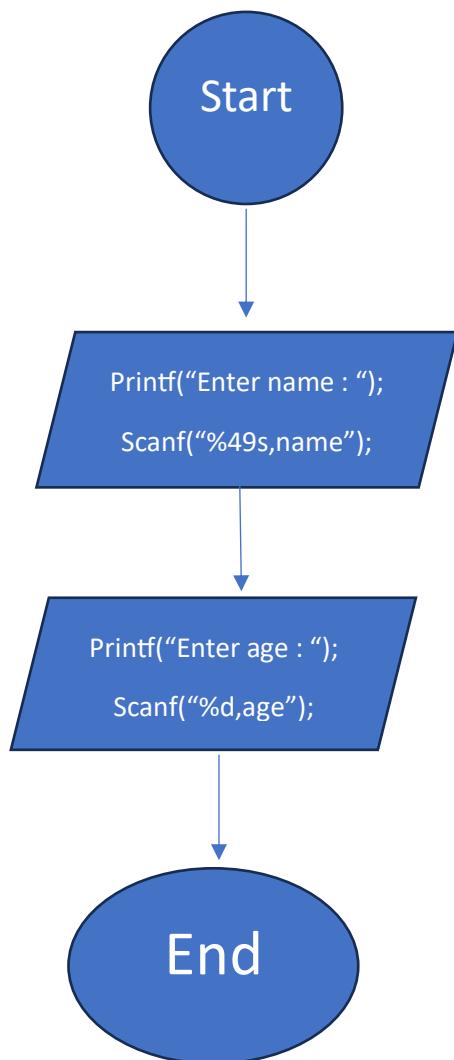
The screenshot shows a terminal window with the following output:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc Name.c -o Name } ; if ($?) { .\Name }
Enter name : Sankalp
Enter age : 18
Hello, Sankalp! You are 18 years old.
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

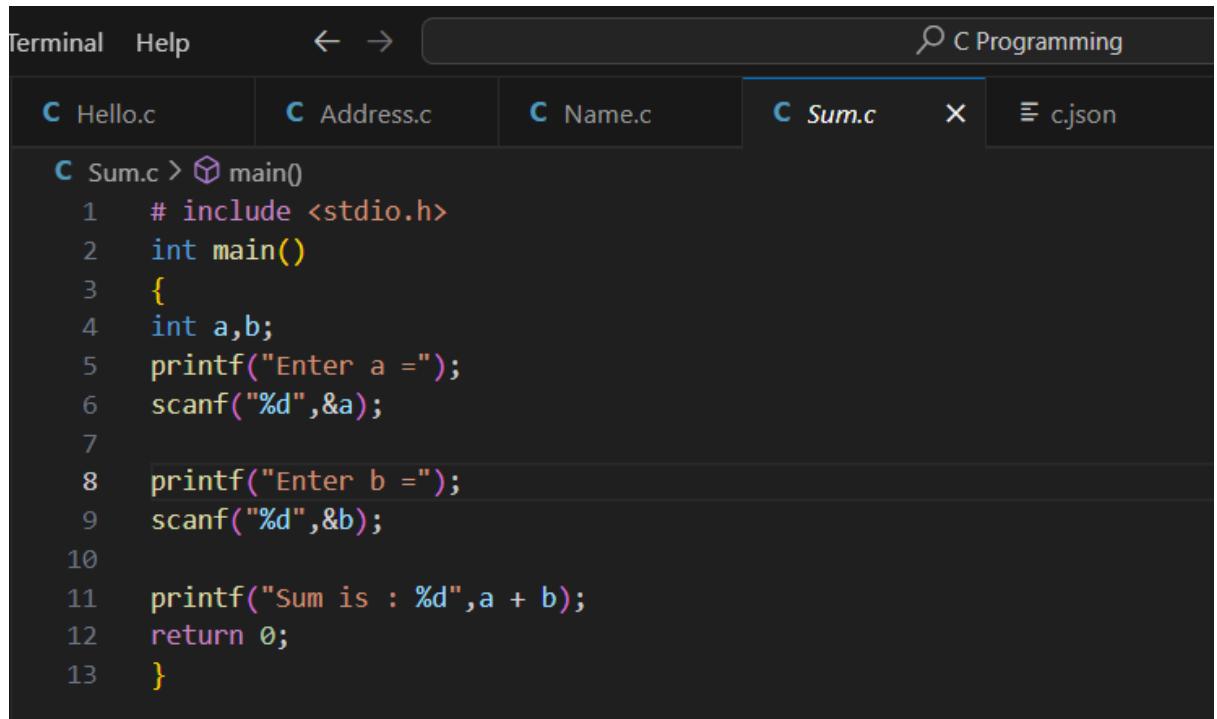
Algorithm :

- 1) Start
- 2) Include header file
- 3) Starting main() function
- 4) Use an input function to ask user to input their name and store it in a string variable.
- 5) Use an input function to ask the user to input their age and store it in an integer variable.
- 6) End

Flow Chart :

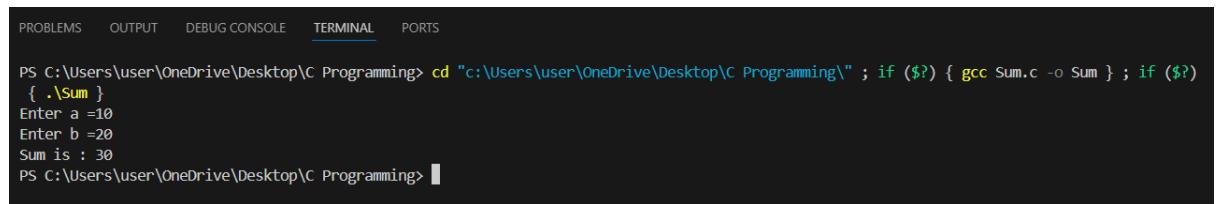


4) add two numbers, take number from user.



```
C Sum.c > main()
1 # include <stdio.h>
2 int main()
3 {
4     int a,b;
5     printf("Enter a =");
6     scanf("%d",&a);
7
8     printf("Enter b =");
9     scanf("%d",&b);
10
11    printf("Sum is : %d",a + b);
12    return 0;
13 }
```

Output :



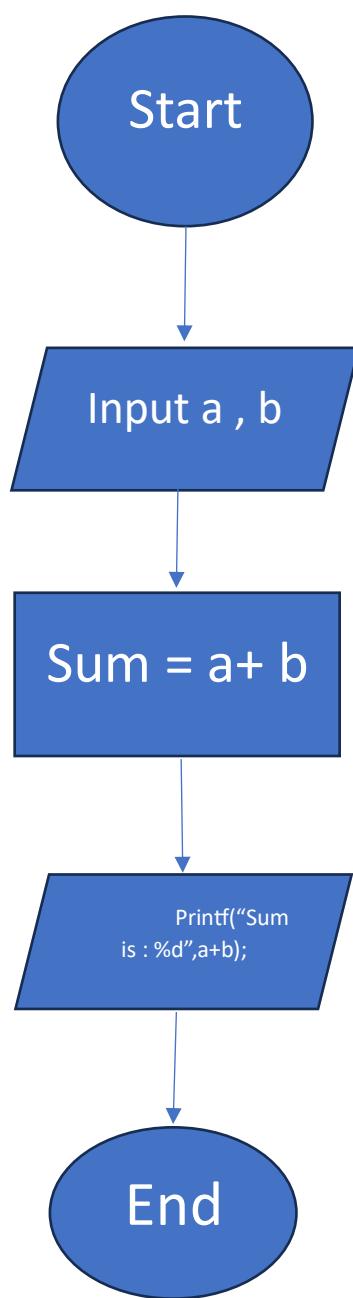
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc Sum.c -o Sum } ; if ($?)
{ .\Sum }
Enter a =10
Enter b =20
Sum is : 30
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

Algorithm :

- 1) Start
- 2) Include header file
- 3) Use two inputs a and b to calculate their sum
- 4) Perform basic calculation as sum = a+b
- 5) End

Flowchart :



5) calculate the area and perimeter of a rectangle based on the input length and width.

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar under 'C PROGRAMMING' containing files like Program5.exe through Program11.exe. The main editor area displays a C program named Rectangle.c:

```
#include <stdio.h>
int main(){
    int l,b;
    printf("Enter length : ");
    scanf("%d",&l);

    printf("Enter breadth : ");
    scanf("%d",&b);

    printf("Area of rectangle : %d\n",l*b);
    printf("Perimeter of rectangle : %d",2*(l+b));
    return 0;
}
```

Output :

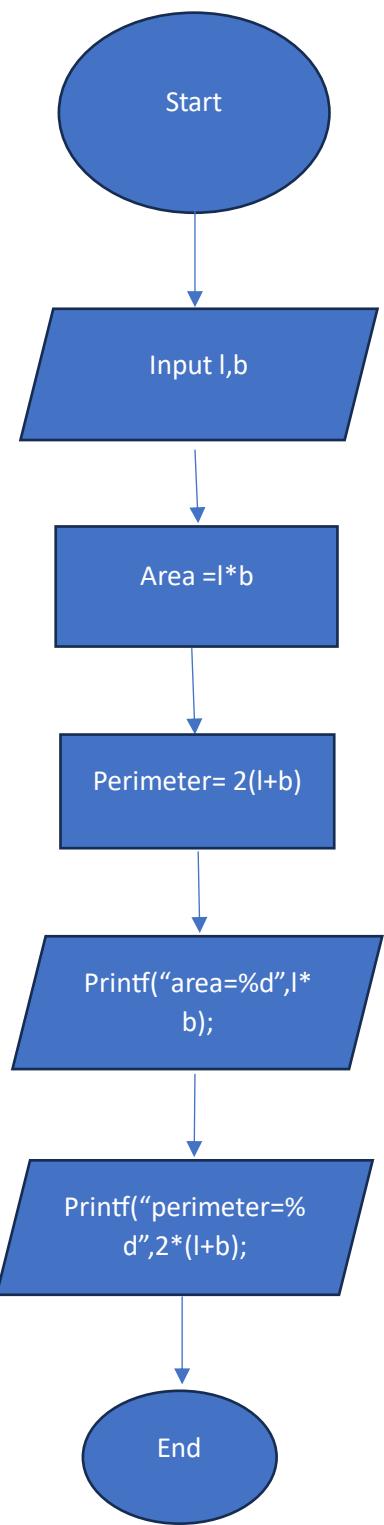
The terminal window shows the execution of the program:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc Rectangle.c -o Rectangle } ; if ($?) {
.\Rectangle }
Enter length : 5
Enter breadth : 6
Area of rectangle : 30
Perimeter of rectangle : 22
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

At the bottom, status information is displayed: Ln 11, Col 44, Spaces: 4, UTF-8, CRLF, { } C, Go Live, Win32, Prettier.

Algorithm :

- 1) Start
- 2) Include header file
- 3) Read length , breadth
- 4) Calculate area = length * breadth
- 5) Calculate perimeter as = 2 * (length + breadth)
- 6) Print area and perimeter
- 7) End



6) convert temperature from Celsius to Fahrenheit
using the formula : $F = (C * 9/5) + 32$

The screenshot shows a code editor interface with a dark theme. At the top, there are tabs for "Terminal", "Help", and several C files: "Hello.c", "Address.c", "Name.c", "Rectangle.c", and "Temperature.c". The "Temperature.c" tab is currently selected. Below the tabs, the code for "Temperature.c" is displayed:

```
C Temperature.c > ↻ main()
1 #include <stdio.h>
2
3 int main(){
4     float celsius,fahrenheit;
5     printf("Enter temperature in celcius : ");
6     scanf("%f",&celsius);
7
8     fahrenheit = (celsius * 9 / 5) + 32;
9
10    printf("Temperature in Fahrenheit: %0.2f\n",fahrenheit);
11    return 0;
12 }
```

Output :

The screenshot shows a terminal window with a dark theme. The tabs at the top are "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", and "PORTS". The "TERMINAL" tab is active. The terminal window displays the following text:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc Temperature.c -o Temperature } ; if ($?
) { .\Temperature }
Enter temperature in celcius : 45
Temperature in Fahrenheit: 113.00
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

At the bottom of the terminal window, there are status indicators: "Ln 10, Col 61", "Spaces: 4", "UTF-8", "CRLF", "C", "Go Live", "Win32", and an empty circle icon.

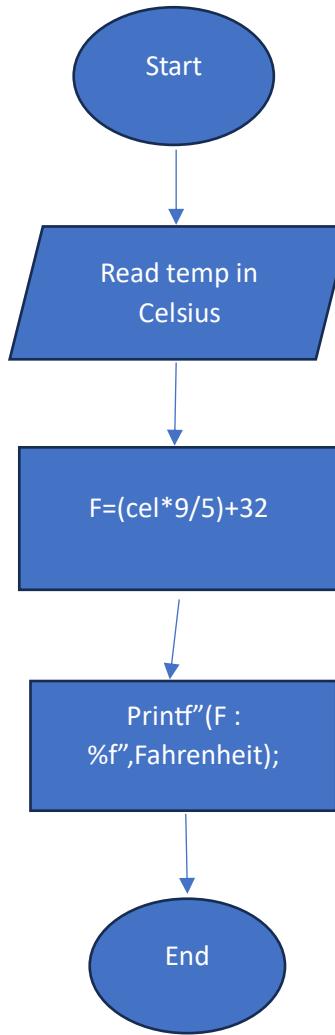
Algorithm :

- 1) Start
- 2) Include header file
- 3) Read temperature in Celsius
- 4) Calculate the temperature from Celsius to Fahrenheit using formula $f=(cel*9/5)+32$

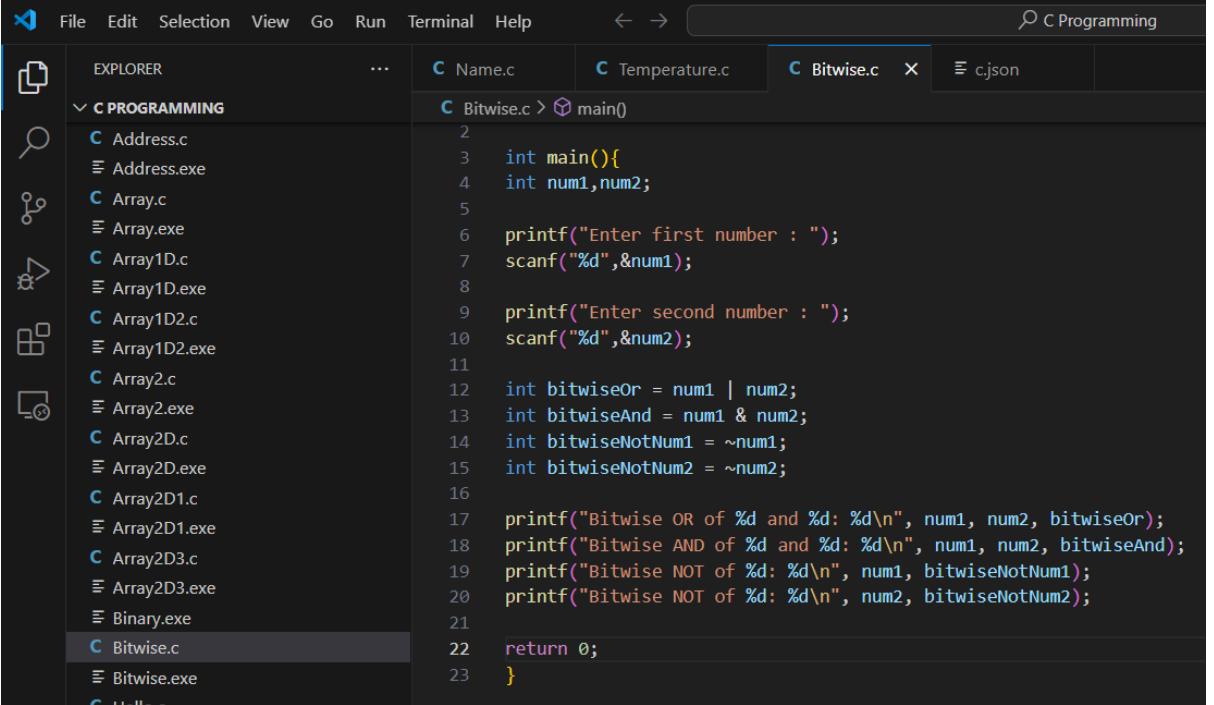
5) Print the temperature in Fahrenheit

6) End

Flowchart :



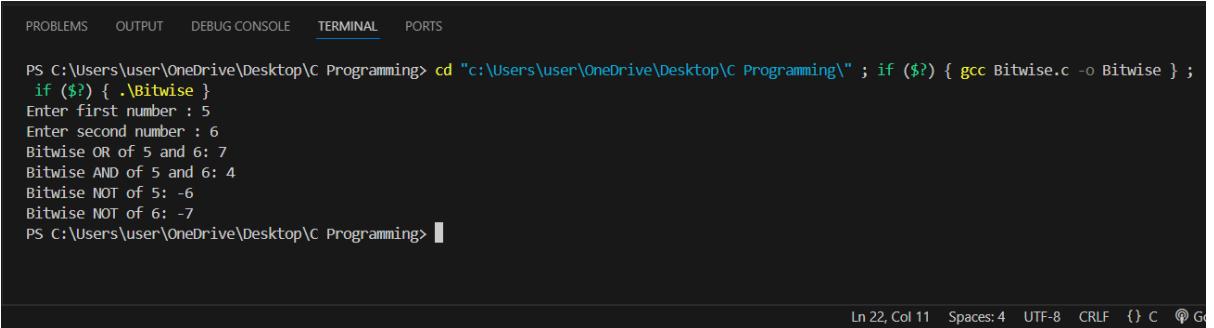
7) Write a C program to apply bitwise OR, AND, and NOT operators on bit level.



The screenshot shows the Visual Studio Code interface. The left sidebar has icons for file operations like Open, Save, Find, and Refresh. The Explorer sidebar shows a tree view under 'C PROGRAMMING' with files like Address.c, Array.c, etc., and a selected 'Bitwise.c'. The main editor area displays the following C code:

```
2 int main(){
3     int num1,num2;
4
5     printf("Enter first number : ");
6     scanf("%d",&num1);
7
8     printf("Enter second number : ");
9     scanf("%d",&num2);
10
11    int bitwiseOr = num1 | num2;
12    int bitwiseAnd = num1 & num2;
13    int bitwiseNotNum1 = ~num1;
14    int bitwiseNotNum2 = ~num2;
15
16    printf("Bitwise OR of %d and %d: %d\n", num1, num2, bitwiseOr);
17    printf("Bitwise AND of %d and %d: %d\n", num1, num2, bitwiseAnd);
18    printf("Bitwise NOT of %d: %d\n", num1, bitwiseNotNum1);
19    printf("Bitwise NOT of %d: %d\n", num2, bitwiseNotNum2);
20
21    return 0;
22 }
```

Output :



The terminal window shows the command to run the program and its output. The user runs 'cd "c:\Users\user\OneDrive\Desktop\C Programming"' followed by 'if (\$?) { gcc Bitwise.c -o Bitwise } ; if (\$?) { .\Bitwise }'. The program prompts for two numbers (5 and 6), performs bitwise operations, and prints the results.

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc Bitwise.c -o Bitwise } ;
if ($?) { .\Bitwise }
Enter first number : 5
Enter second number : 6
Bitwise OR of 5 and 6: 7
Bitwise AND of 5 and 6: 4
Bitwise NOT of 5: -6
Bitwise NOT of 6: -7
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

At the bottom right of the terminal, there are status indicators: Ln 22, Col 11, Spaces: 4, UTF-8, CRLF, {}, C, and a gear icon.

Algorithm :

8) Write a C program to apply left and right shift operators.

The screenshot shows the Visual Studio Code interface. The left sidebar has a tree view under 'C PROGRAMMING' containing files like Program7.exe through Program16.c. The main editor area shows the 'Shift.c' file with the following code:

```
#include <stdio.h>
int main() {
    int num, shiftAmount;
    printf("Enter an integer: ");
    scanf("%d", &num);

    printf("Enter the number of positions to shift: ");
    scanf("%d", &shiftAmount);

    int leftShift = num << shiftAmount;
    int rightShift = num >> shiftAmount;

    printf("Original number: %d\n", num);
    printf("Left shift by %d positions: %d\n", shiftAmount, leftShift);
    printf("Right shift by %d positions: %d\n", shiftAmount, rightShift);

    return 0;
}
```

Output :

The screenshot shows the terminal tab in VS Code with the following session:

```
Bitwise OR of 5 and 6: 7
Bitwise AND of 5 and 6: 4
Bitwise NOT of 5: -6
Bitwise NOT of 6: -7
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc Shift.c -o shift } ; if
($) { .\shift }
Enter an integer: 8
Enter the number of positions to shift: 4
Original number: 8
Left shift by 4 positions: 128
Right shift by 4 positions: 0
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

At the bottom of the terminal window, status indicators show: Ln 7, Col 23, Spaces: 4, UTF-8, CRLF, {}, C, ⌂ G.

Experiment : Conditional Statements

1. WAP to take check if the triangle is valid or not. If the validity is established, do check if the triangle is isosceles, equilateral, right angle, or scalene. Take sides of the triangle as input from a user.

```
C Name.c      C Transpose.c      C Triangle.c X      C Bitwise.c      c.json
C Triangle.c > ② main()
1 #include <stdio.h>
2
3 int main(){
4     float a,b,c;
5
6     printf("Enter length of side A: ");
7     scanf("%f",&a);
8     printf("Enter length of side B: ");
9     scanf("%f",&b);
10    printf("Enter length of side C: ");
11    scanf("%f",&c);
12
13    if((a+b>c)&& (a+c>b)&& (b+c>a)){
14        printf("The triangle is valid\n");
15
16        if(a==b && b==c){
17            printf("The traingle is equilateral\n");
18        }
19        else if(a==b || b==c || a==c){
20            printf("The traingle is isosceles\n");
21        }
22        else{
23            printf("The triangle is scalene\n");
24        }
25    }
26 }
```

Output :

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

The triangle is valid
The triangle is scalene
The triangle is also a right triangle
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc Triangle.c -o Triangle }
; if ($?) { ./Triangle }
Enter length of side A: 5
Enter length of side B: 5
Enter length of side C: 5
The triangle is valid
The traingle is equilateral
The triangle is not a right triangle
PS C:\Users\user\OneDrive\Desktop\C Programming> 
```

Ln 20, Col 51 Spaces: 4 UTF-8 CRLF {} C Go Live W

Step 1: Declare three variables a,b,c of triangle.

Step 2: Enter three sides .

Step 3: If $a == b \&\& b == c$ Go to step 6

Step 4: If $a == b || b == c || c == a$ Go to Step 7

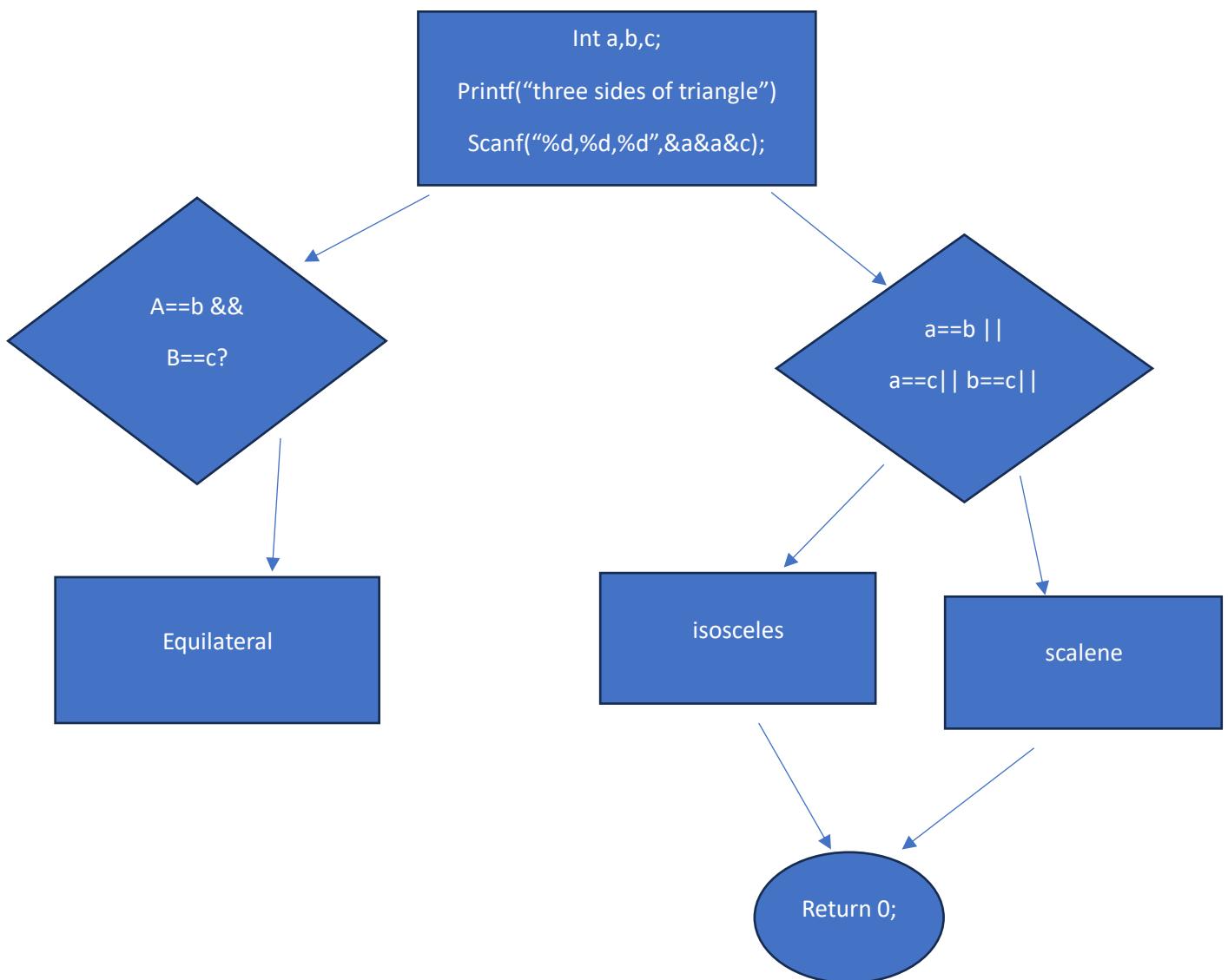
Step 5: else Go to step 8

Step 6: Print the triangle is equilateral.

Step 7: Print the triangle is isosceles.

Step 8: Print the triangle is scalene.

Flowchart :

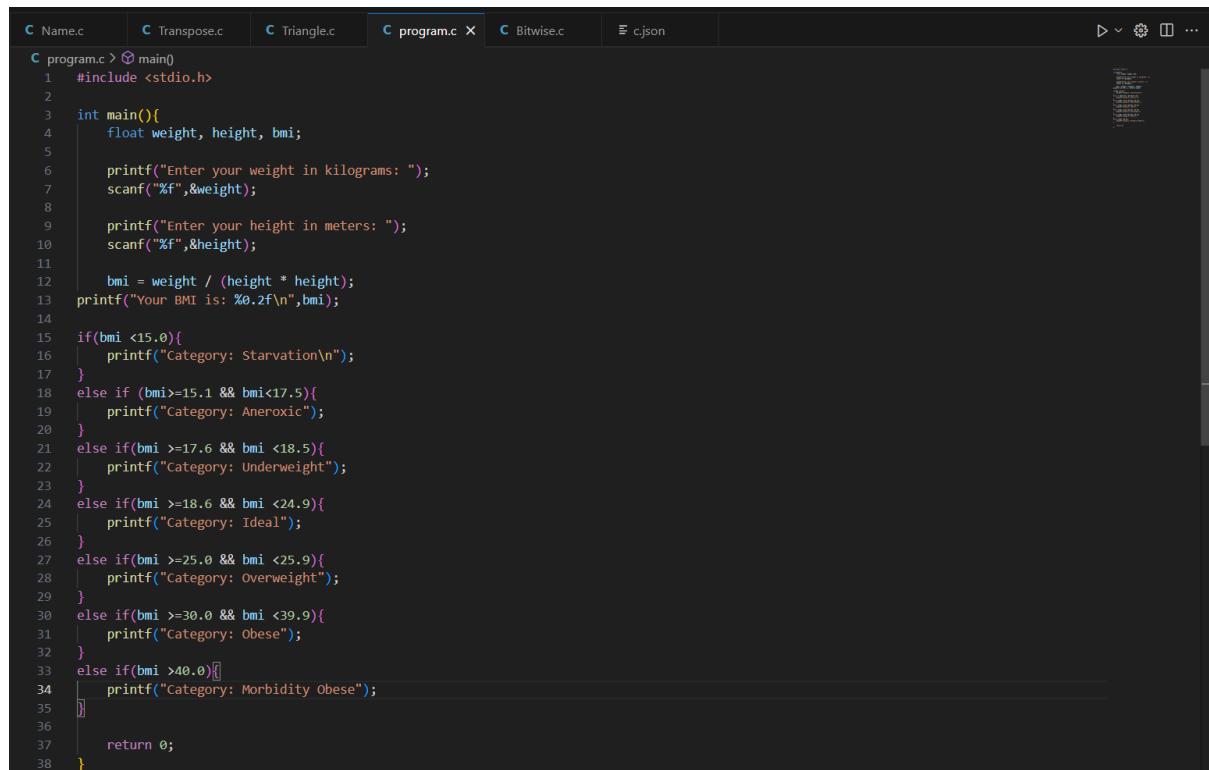


2. WAP to compute the BMI Index of the person and print the BMI values as per the following ranges.

You can use the following formula to compute
 $BMI = \text{weight(kgs)}/\text{Height(Mts)} * \text{Height(Mts)}$. BMI

Starvation is

	BMI
Starvation	<15
Anorexic	15.1 to 17.5
Underweight	17.6 to 18.5
Ideal	18.6 to 24.9
Overweight	25 to 25.9
Obese	30 to 39.9
Morbidity Obese	40 above



The screenshot shows a code editor window with multiple tabs at the top: Name.c, Transpose.c, Triangle.c, program.c (which is the active tab), Bitwise.c, and cJSON. The code in the 'program.c' tab is as follows:

```
#include <stdio.h>
int main(){
    float weight, height, bmi;
    printf("Enter your weight in kilograms: ");
    scanf("%f",&weight);
    printf("Enter your height in meters: ");
    scanf("%f",&height);
    bmi = weight / (height * height);
    printf("Your BMI is: %.2f\n",bmi);
    if(bmi <15.0){
        printf("Category: Starvation\n");
    }
    else if (bmi>=15.1 && bmi<17.5){
        printf("Category: Anorexic");
    }
    else if(bmi >=17.6 && bmi <18.5){
        printf("Category: Underweight");
    }
    else if(bmi >=18.6 && bmi <24.9){
        printf("Category: Ideal");
    }
    else if(bmi >=25.0 && bmi <25.9){
        printf("Category: Overweight");
    }
    else if(bmi >=30.0 && bmi <39.9){
        printf("Category: obese");
    }
    else if(bmi >40.0){
        printf("Category: Morbidity obese");
    }
    return 0;
}
```

Output :

The screenshot shows a terminal window with the following content:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc program.c -o program } ; if ($?) { ./program }
Enter your weight in kilograms: 75.6
Enter your height in meters: 5.2
Your BMI is: 2.80
Category: Starvation

PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc program.c -o program } ; if ($?) { ./program }
Enter your weight in kilograms: 74.5
Enter your height in meters: 1.74
Your BMI is: 24.61
Category: Ideal

PS C:\Users\user\OneDrive\Desktop\C Programming>
```

At the bottom of the terminal window, there are status indicators: Line 34, Col 41, Spaces: 4, UTF-8, CRLF, a file icon, Go Live, Win32, Prettier, and a refresh icon.

Algorithm :

Step 1:Start

Step2:input weight in kgs and height in meters(in float)

Step3:use bmi formula : $bmi = \frac{weight}{height^2}$

Step4: check conditions If $bmi \leq 15$ print “starvation”

Else if $15.1 \leq bmi \leq 17.5$ print “anorexic” Else if $17.6 \leq bmi \leq 18.5$

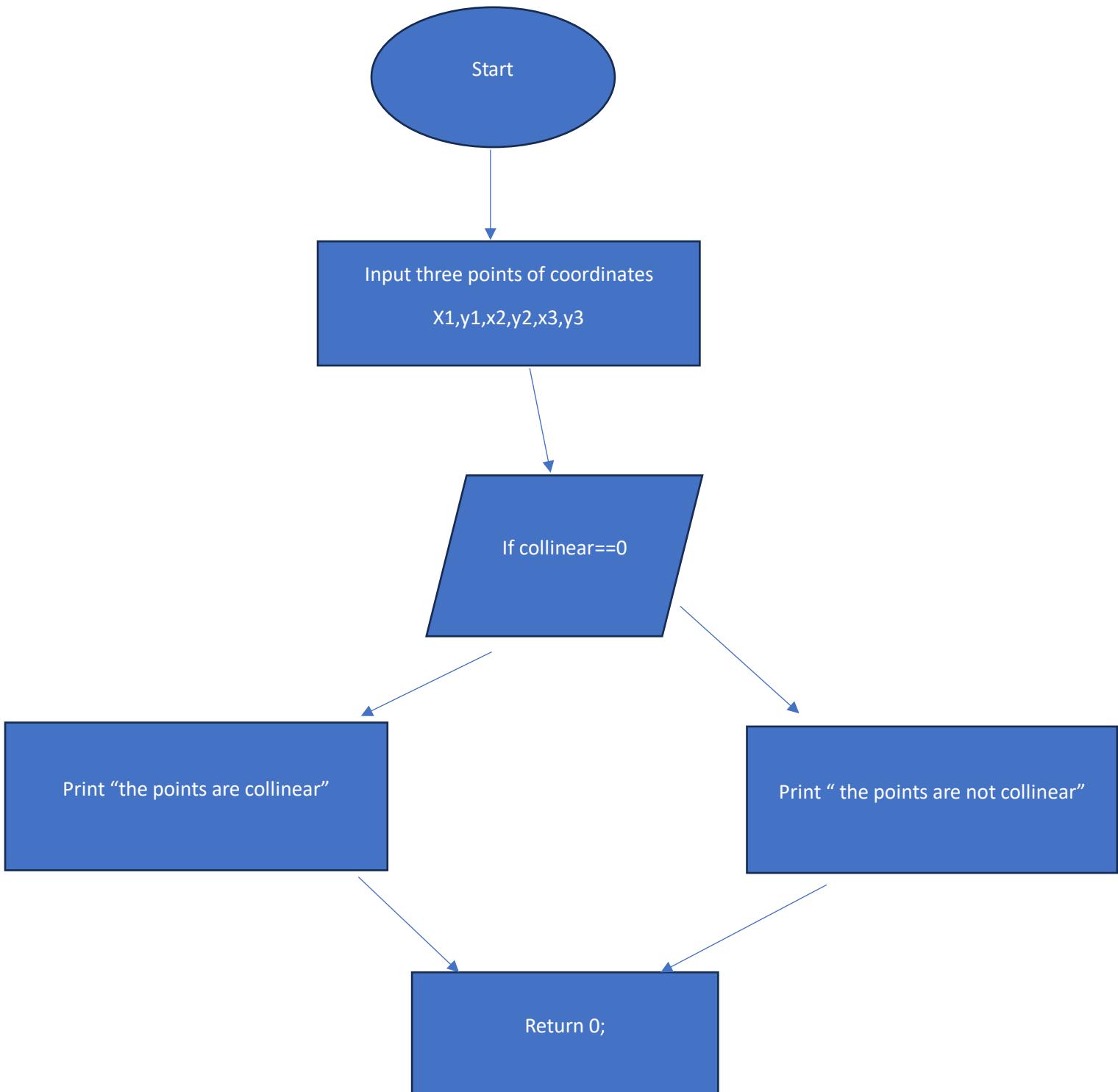
print “underweight” Else if $18.6 \leq bmi \leq 24.9$ print “ideal” Else

if $25 \leq bmi \leq 25.9$ print “overweight” Else if $30 \leq bmi \leq 39.9$ print “obese” Else Print “Morbidity Obese”

Step 5-display the bmi category which matches the condition .

Step6-End

Flowchart :



3. WAP to check if three points (x_1, y_1) , (x_2, y_2) and (x_3, y_3) are collinear or not.

```
C Name.c | C Transpose.c | C Triangle.c | C Temperature.c | C program.c | C Collinear.c X | C Bitwise.c | E cJSON
C Collinear.c > ⌂ main()
1 #include <stdio.h>
2
3 int main(){
4     float x1,y1,x2,y2,x3,y3;
5
6     printf("Enter coordinates of point 1 (x1, y1): ");
7     scanf("%f%f",&x1, &y1);
8     printf("Enter coordinates of point 2 (x2, y2): ");
9     scanf("%f%f",&x2, &y2);
10    printf("Enter coordinates of point 3 (x3, y3): ");
11    scanf("%f%f",&x3, &y3);
12
13    float area = x1 * (y2 - y3) + x2 * (y3 - y1) + x3 * (y1 - y2);
14    printf("Area = %.2f\n",area);
15
16    if(area == 0){
17        printf("The points are collinear\n");
18    }
19    else{
20        printf("The points are not collinear\n");
21    }
22    return 0;
23 }
```

Output :

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code + ⌂ ⌂ ⌂ ⌂ ⌂ ⌂  
Area = 0.00 The points are collinear  
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc Collinear.c -o Collinear } ; if ($?) { .\Collinear  
}  
Enter coordinates of point 1 (x1, y1): 2  
3  
Enter coordinates of point 2 (x2, y2): 4  
5  
Enter coordinates of point 3 (x3, y3): 6  
7  
Area = 0.00  
The points are collinear  
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

Algorithm :

Step1-Start

Step2-input all three points of coordinates

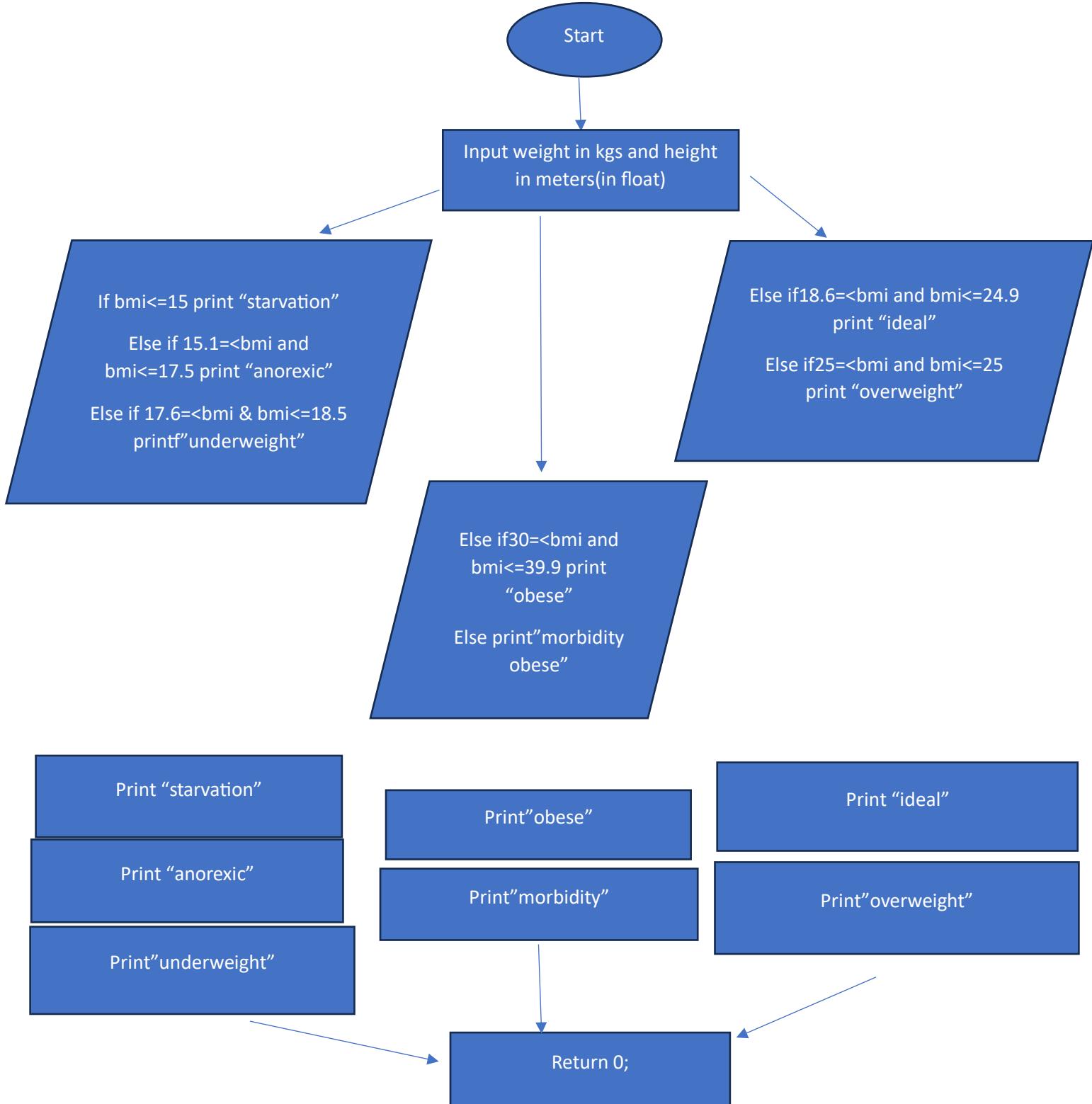
x1,y1,x2,y2,x3,y3

Step3-use formula – $((x_1 * (y_2 - y_3)) + x_2 * (y_3 - y_1) + x_3 * (y_1 - y_2))$

Step4-check condition If($\text{collinear}==0$) Print “points are collinear” Else Print “the points are not collinear”

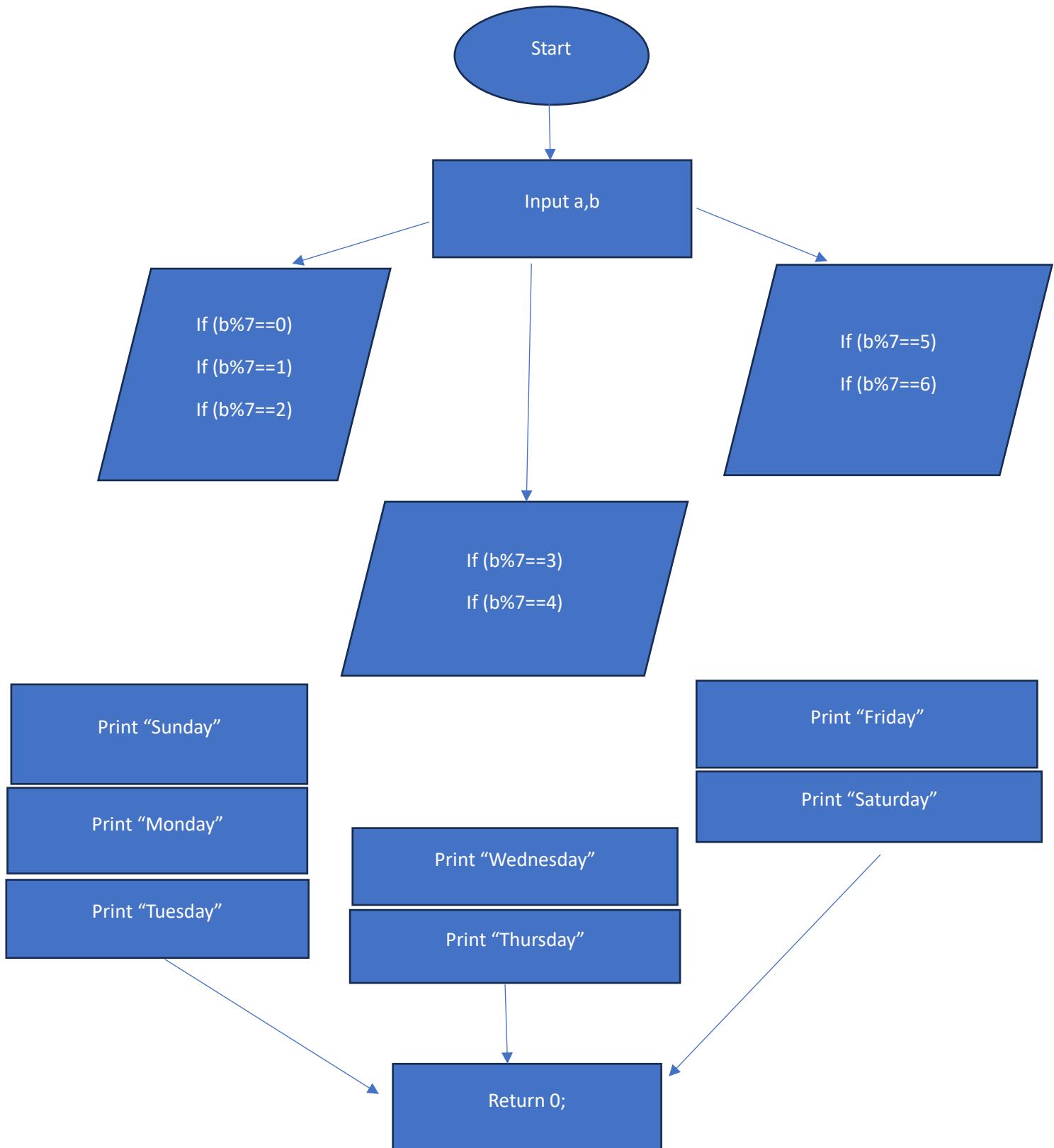
Step5- End

Flowchart :



4. According to the gregorian calendar, it was Monday on the date 01/01/01. If Any year is input through the keyboard write a program to find out what is the day on 1st January of this year.

Flowchart :



C Name.c C Transpose.c C Triangle.c C program.c C Collinear.c C Ternary.c C calendar.c X C Bitwise.c E c.json D v S

```
calendar.c > main()
1 #include <stdio.h>
2
3 int main()
4 {
5     int yearGiven = 2001;
6
7     int year;
8     printf("Enter the year: ");
9     scanf("%d",&year);
10
11    int difference = year - yearGiven;
12
13    int leapYear = difference/4;
14
15    int nonLeapYear = difference - leapYear;
16
17    int days = (leapYear * 366) + (nonLeapYear * 365) +1;
18
19    if(days % 7 ==0)
20    {
21        printf("Sunday");
22    }
23    if(days % 7 ==1)
24    {
25        printf("Monday");
26    }
27    if(days % 7 ==2)
28    {
29        printf("Tuesday");
30    }
31    if(days % 7 ==3)
32    {
33        printf("Wednesday");
34    }
35    if(days % 7 ==4)
36    {
37        printf("Thursday");
38    }
39    if(days % 7 ==5)
40    {
41        printf("Friday");
42    }
43    if(days % 7 ==6)
44    {
45        printf("Saturday");
46    }
47 }
```

Output :

Algorithm :

Step1-Start

Step2-input a, b (a for year ,b to calculating, what is the day on 1st January of this year)

Step3-use formula –) $b=(a+(a/4)-(a/100)+(a/400))\%7$

Step4-check condition If($b\%7==0$) Print” Sunday”

If($b\%7==1$)Print” Monday” If($b\%7==2$)print” Tuesday”

If($b\%7==3$)print” Wednesday” If($b\%7==4$)print

“Thursday” If($b\%7==5$)print “Friday” If($b\%7==6$)print

“Saturday”

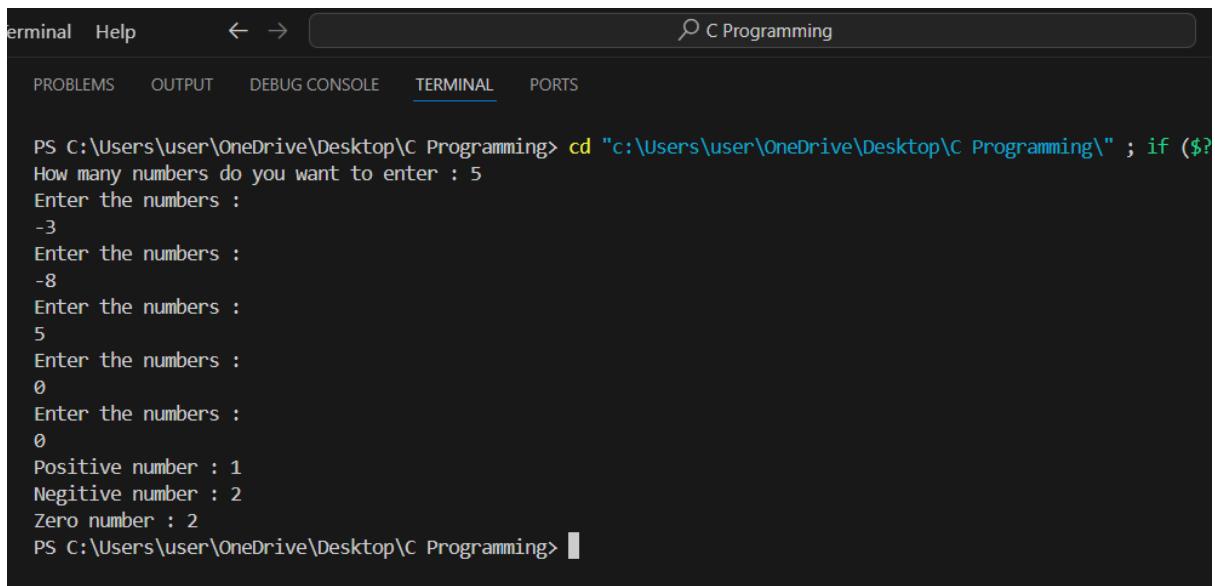
Step5- End

Experiment : Loops

1. WAP to enter numbers till the user wants. At the end, it should display the count of positive, negative, and Zeroes entered.

```
C Name.c    C Collinear.c    C calendar.c    C count.c    X C Bitwise.c    E cJSON
C count.c > main()
3 int main(){
6     printf("How many numbers do you want to enter : ");
7     scanf("%d",&a);
8
9     int positive = 0;
10    int negative = 0;
11    int zero = 0;
12
13    int i = 1;
14    while(i <= a)
15    {
16        int number;
17        printf("Enter the numbers : \n");
18        scanf("%d",&number);
19
20        if(number == 0)
21        {
22            zero++;
23        }
24        else if(number > 0)
25        {
26            positive++;
27        }
28        else if(number < 0)
29        {
30            negative++;
31        }
32        i++;
33    }
34
35    printf("Positive number : %d\n",positive);
36    printf("Negative number : %d\n",negative);
37    printf("Zero number : %d\n",zero);
38
39    return 0;
}
```

Output :



A screenshot of a terminal window titled "C Programming". The window has tabs for "TERMINAL" (which is selected) and "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", and "PORTS". The terminal output shows a command-line session:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if (${?}
How many numbers do you want to enter : 5
Enter the numbers :
-3
Enter the numbers :
-8
Enter the numbers :
5
Enter the numbers :
0
Enter the numbers :
0
Positive number : 1
Negative number : 2
Zero number : 2
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

Algorithm :

Step1-Start

Step2-input a for how many number you want to enter

Step3-take Initialize counters: positive=0 negative =0 zero =0

Step4-start a loop for accepting numbers

Step5-now input the numbers inside loop

Step6-check condition

if number>0 then positive ++;

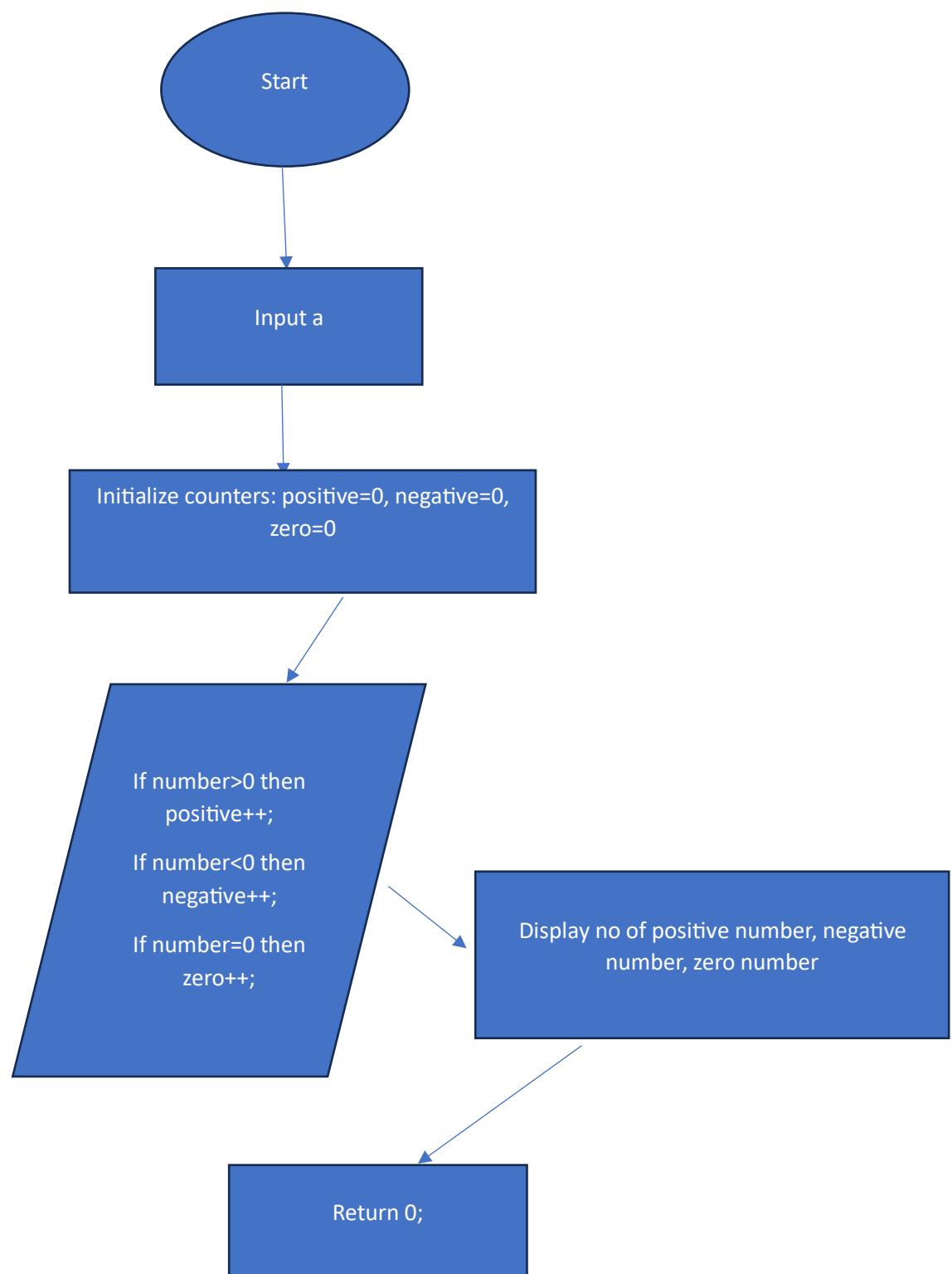
if number<0 then negative++;

if number=0 then zero++;

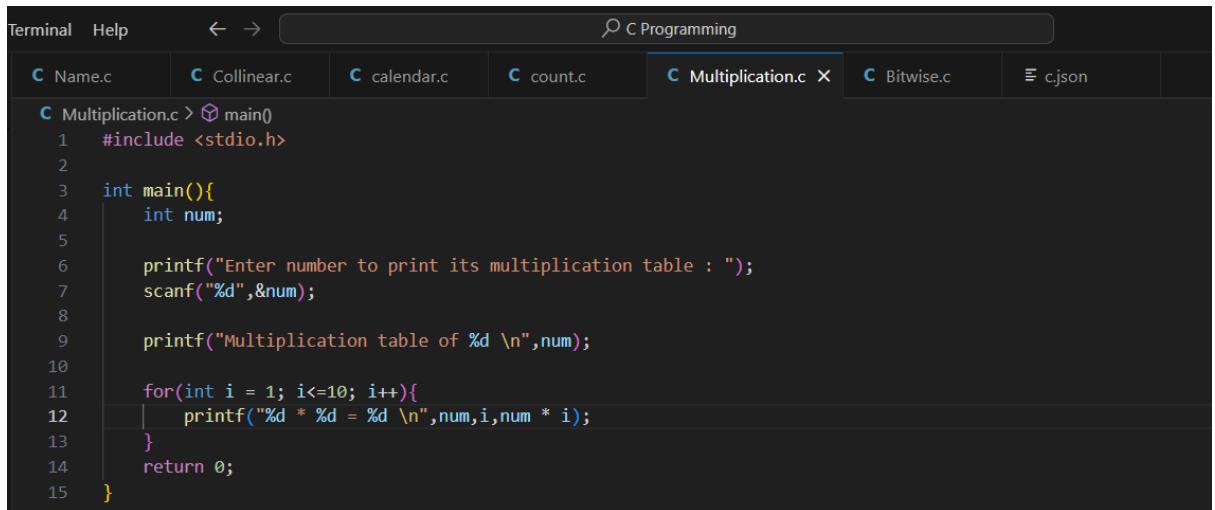
Step-8 now display the output after loop ends it will give you number of positive number negative numbers and number of zeros in the code

Step-9-End

Flowchart :

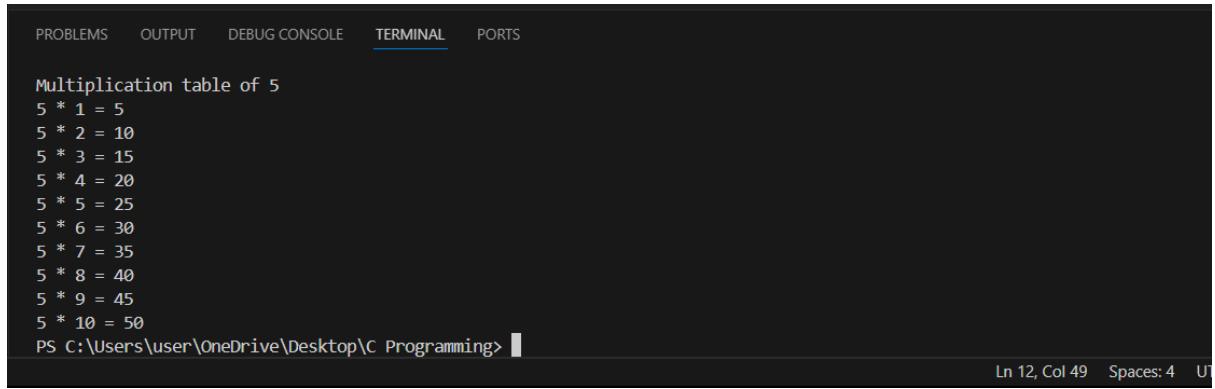


WAP to print the multiplication table of the number entered by the user. It should be in the correct formatting. Num * 1 = Num.



```
Terminal Help ← → C Programming
C Name.c C Collinear.c C calendar.c C count.c C Multiplication.c X C Bitwise.c c.json
C Multiplication.c > main()
1 #include <stdio.h>
2
3 int main(){
4     int num;
5
6     printf("Enter number to print its multiplication table : ");
7     scanf("%d",&num);
8
9     printf("Multiplication table of %d \n",num);
10
11    for(int i = 1; i<=10; i++){
12        printf("%d * %d = %d \n",num,i,num * i);
13    }
14
15 }
```

Output :



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Multiplication table of 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
PS C:\Users\user\OneDrive\Desktop\C Programming> Ln 12, Col 49 Spaces: 4 U
```

Algorithm :

Step1-Start

Step2-input a number

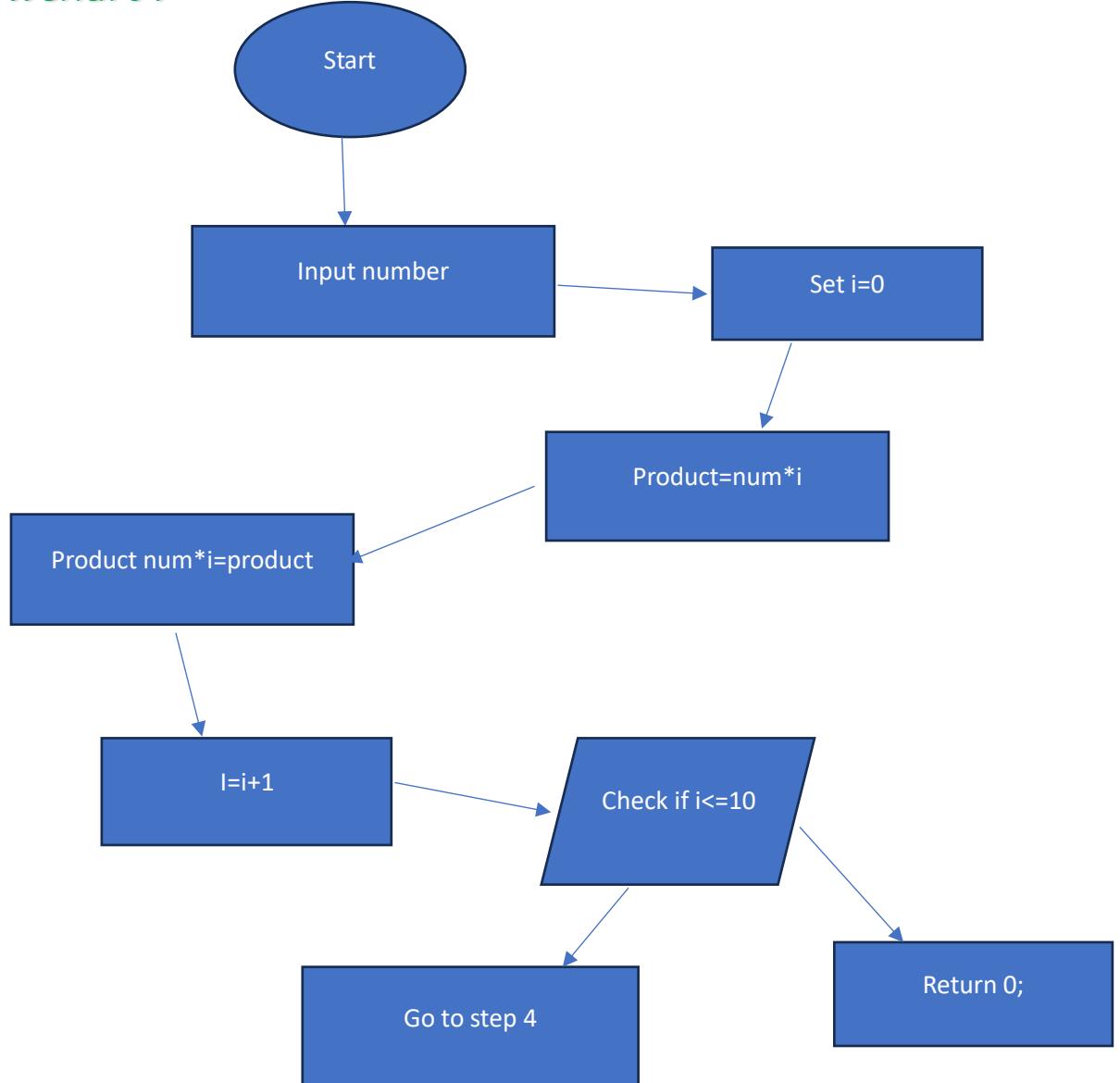
Step3-initialize a loop from 1 to 10

Step4-Multiply the number: For each iteration of the loop (from 1 to 10), calculate the product of the number (num) and the current iteration index (i).

Step5-Display the result in the format: num * i = product.

Step 6- End

Flowchart :



2. WAP to generate the following patterns (we can done 7 of them in lectures, finish all):

Terminal Help ← → 🔍 C Programming

count.c | C Multiplication.c | C Program16.c | C Program9.c | C Program8.c | C Practise3.c | C Practice9.c | C Pr

```
C Practice7.c > ↻ main()
1 # include<stdio.h>
2 int main()
3 {
4     int c=1;
5     for (int i =1; i <=5; i++){
6         for(int j =1; j <=i; j++){
7             printf("%d",c);
8             c++;
9         }
10        printf("\n");
11    }
12    return 0;
13 }
```

1.

Output :

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if (\$?) { gcc Practice7.c -o Practice7 } ; if (\$?) { .\Practice7 }

```
1
23
456
78910
1112131415
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

Ln 10, Col 22 Spaces:4 UTF-8 CRLF {} C ⚙ Go Live Win32 ⚙ Pretty

Terminal Help ← → 🔍 C Programming

calendar.c | C count.c | C Multiplication.c | C Program16.c | C Program9.c | C Program8.c | C Practise3.c | C Practice9.c | C Pr

```
C Practice8.c > ↻ main()
1 # include<stdio.h>
2 int main()
3 {
4     int i,j,k;
5     for(i=1; i<=5; i++)
6     {
7         for(j=i;j<5; j++){
8             printf(" ");
9         }
10        for(k=1; k<=i; k++){
11            printf("**");
12        }
13        printf("\n");
14    }
15    return 0;
16 }
```

2.

Output :

The screenshot shows a terminal window with the following text:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc Practice8.c -o Practice8
.\Practice8 }
*
**
***
****
*****
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

At the bottom right of the terminal window, it says "Ln 12, Col 10 Spaces: 4 UTF-8 CRLF {} C ⌂".

The screenshot shows a code editor with the following code in the file Practise3.c:

```
C Practise3.c > main()
1 # include<stdio.h>
2 int main()
3 {
4     int i,j,rows;
5     printf("Enter number of rows : ");
6     scanf("%d",&rows);
7
8     for(i=rows; i>=1; i--){
9         for(j=1; j<=i; j++){
10            printf("*");
11        }
12        printf("\n");
13    }
14
15 }
```

At the top of the code editor, there is a navigation bar with tabs for Name.c, Collinear.c, calendar.c, count.c, Multiplication.c, Program16.c, Program9.c, and Program17.c. The title bar says "C Programming".

Output :

The screenshot shows a terminal window with the following text:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc Practise3.c -o Practise3
.\Practise3 }
Enter number of rows : 5
*****
*****
****
 ***
 *
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

At the bottom right of the terminal window, it says "Ln 9, Col 29 Spaces: 4 UTF-8 CRLF {} C ⌂ G".

4.

```
C Name.c C Collinear.c C calendar.c C count.c C Multiplication.c C Program16.c C Program11.c X C Bitwise.c
C Program11.c > main()
1 # include <stdio.h>
2 int main()
3 {
4     int rows = 5;
5
6     for (int i = 0; i < rows; i++) {
7
8         for (int j = 0; j <= i; j++) {
9             printf("* ");
10        }
11        printf("\n");
12    }
13    return 0;
14 }
```

Output :

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc Program11.c -o Program11 } ; .\Program11
*
* *
* * *
* * * *
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

Ln 10, Col 10 Spaces: 4 UTF-8 CRLF {} C ⌂ Go L

5.

```
Terminal Help ← → C Programming
C book.c ● C pattern1.c X c.json
C pattern1.c > main()
1 #include <stdio.h>
2
3 int main(){
4     int n=1;
5     for(int i=1; i<=3; i++) {
6         for(int j=1; j<=i; j++) {
7             printf(" ");
8         }
9     }
10    for(int h =1; h<=(2 * i -1); h++) {
11        printf("%d",n);
12        n++;
13    }
14    printf("\n");
15 }
16 }
```

Output :

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming"
1
234
56789
PS C:\Users\user\OneDrive\Desktop\C Programming>

Ln 12, Col 17  Spaces: 4  UTF-8  CRLF  {} C ⌂
```

6.

```
Terminal Help ← → C Programming
.. C book.c ● C pattern1.c C pattern2.c X c.json
C pattern2.c > ...
1 #include <stdio.h>
2
3 int main(){
4     int a,b,c=1,x,n;
5     printf("enter no of rows :");
6     scanf("%d",&n);
7     for(x=0; x<=n-1; x++) {
8         for(b=0; b<=n-x; b++)
9             printf(" ");
10            for(a=0; a<=x; a++)
11            {
12                if(a==0 || x==0)
13                    c=1;
14                else
15                    c=c*(x-a+1)/a;
16                printf("%4d",c);
17            }
18        printf("\n");
19    }
20}
21
22
```

Output :

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc pattern2.c -o pattern2 } ; if
enter no of rows :5
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
PS C:\Users\user\OneDrive\Desktop\C Programming>

Ln 22, Col 1 Spaces: 4 UTF-8 CRLF {} C Go Live
```

Q15- The population of a town is 100000. The population has increased steadily at the rate of 10% per year for the last 10 years. Write a program to determine the population at the end of each year in the last decade.

```
Terminal Help ← → C Programming
C book.c ● C pattern1.c C pattern2.c C file1.c X c.json
C file1.c > main()
1 #include <stdio.h>
2
3 int main()
4 {
5     float b;
6     int n = 100000;
7     printf("Population at the end of 10 years was %d\n",n);
8     float a =0.1;
9     for(int i = 10; i>0; i--)
10    {
11        n=n-a*n;
12        printf("Population last year was %d\n",n);
13    }
14    return 0;
15 }
```

Output :

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Population at the end of 10 years was 100000
Population last year was 89999
Population last year was 80999
Population last year was 72899
Population last year was 65609
Population last year was 59048
Population last year was 53143
Population last year was 47828
Population last year was 43045
Population last year was 38740
Population last year was 34865
PS C:\Users\user\Desktop\C Programming> 
```

Algorithm :

Step1-Start

Step 2- Initialize the population n to 100,000.

Initialize the decline rate a to 0.1 (10%).

Step 3- Print the initial population n using printf

Step 4- Loop 10 times, decrementing the loop counter i from 10 to 1

Step 5-End

Q16- Ramanujan Number is the smallest number that can be expressed as the sum of two cubes in two different ways. WAP to print all such numbers up to a reasonable limit. Example of Ramanujan number: $1729 = 12^3 + 1^3$ and $1729 = 10^3 + 9^3$. for a number L=20(that is limit)

The screenshot shows a dark-themed IDE interface with a tab bar at the top. The active tab is 'file2.c'. Other tabs include 'book.c', 'pattern1.c', 'pattern2.c', 'file1.c', and 'c.json'. The code editor displays the following C program:

```
#include <stdio.h>
int main(){
    int a=0;
    for(int i=0; i<35000; i++) {
        a=0;
        for(int j=1; (j*j*j)<=i; j++){
            for(int k=j; (j*j*j)+(k*k*k)<=i; k++) {
                if(j*j*j+k*k*k==i){
                    a++;
                    if(a==2) {
                        printf("ramanujan number is %d\n",i);
                    }
                }
            }
        }
    }
    return 0;
}
```

Output :

The terminal window shows the command being run and its output. The command is:

```
cd "c:\Users\user\OneDrive\Desktop\C Programming" ; if ($?) { gcc file2.c -o file2 ; ./file2 }
```

The output shows several Ramanujan numbers found:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming" ; if ($?) { gcc file2.c -o file2 ; ./file2 }
ramanujan number is 1729
ramanujan number is 4104
ramanujan number is 13832
ramanujan number is 20683
ramanujan number is 32832
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

At the bottom right of the terminal window, there is status information: Ln 17, Col 6 Spaces: 4 UTF-8 CRLF.

Algorithm :

Step1- Input a=0;

Step2-Initialize loop limit from 0 to 35000

Step 3-Generate Sums of Cubes in i j k using loop

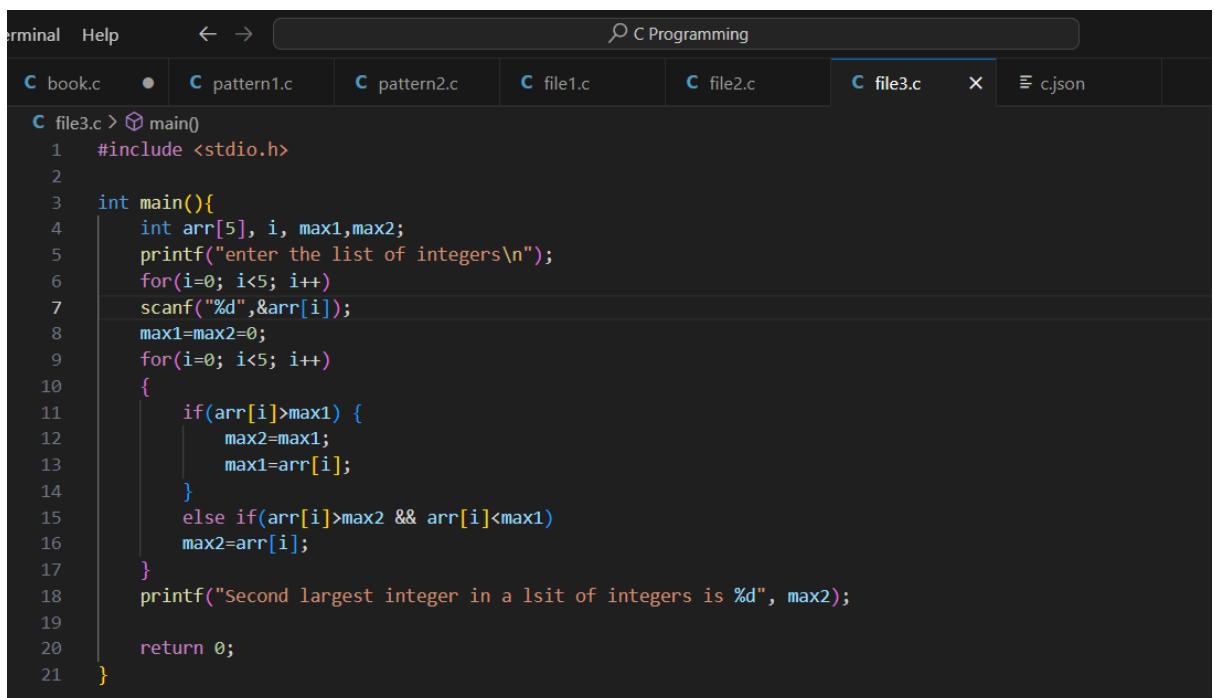
Step 4-check ramanujan number

Step5-Print ramanujan number

Step6-End

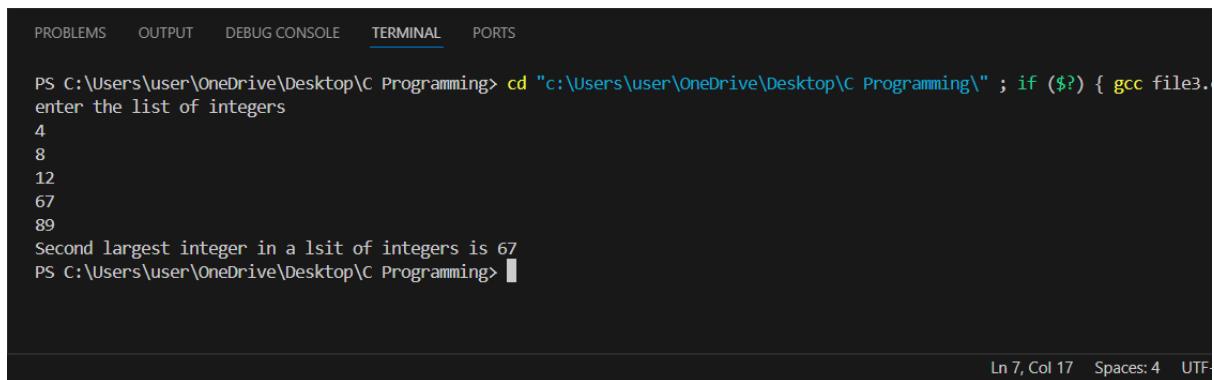
Experiment- Arrays

Q17- WAP to read a list of integers and store it in a 1D array. Write a C program to print the second largest integer in a list of integers.



```
Terminal Help ← → 🔍 C Programming
C book.c ● C pattern1.c C pattern2.c C file1.c C file2.c C file3.c X c.json
C file3.c > ↗ main()
1 #include <stdio.h>
2
3 int main(){
4     int arr[5], i, max1,max2;
5     printf("enter the list of integers\n");
6     for(i=0; i<5; i++)
7         scanf("%d",&arr[i]);
8     max1=max2=0;
9     for(i=0; i<5; i++)
10    {
11        if(arr[i]>max1) {
12            max2=max1;
13            max1=arr[i];
14        }
15        else if(arr[i]>max2 && arr[i]<max1)
16            max2=arr[i];
17    }
18    printf("Second largest integer in a lsit of integers is %d", max2);
19
20    return 0;
21 }
```

Output :



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc file3.c ; ./file3 }
enter the list of integers
4
8
12
67
89
Second largest integer in a lsit of integers is 67
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

Q18- WAP to read a list of integers and store it in a 1D array. Write a C program to count and display positive, negative, odd, and even numbers in an array.

The screenshot shows a C programming environment with the following details:

- Terminal:** Help
- File Tabs:** book.c (selected), pattern1.c, pattern2.c, file1.c, file2.c
- Search Bar:** C Programming
- Code Area:** The code is named file4.c and contains the following C code:

```
C file4.c > main()
1 #include <stdio.h>
2
3 int main(){
4     int a[6],i,p=0,n=0,m=0,c=0;
5     printf("\n enter the list of integers : ");
6     for(i=0; i<6; i++)
7         scanf("%d",&a[i]);
8
9     for(i=0; i<6; i++)
10    {
11        if(a[i]%2 ==0)
12            c++;
13    }
14    printf("\neven numbers count%d",c);
15    for(i=0; i<6; i++) {
16        if(a[i] %2 !=0)
17            m++;
18    }
19    printf("\nodd numbers count%d",m);
20
21    for(i=0; i<6; i++) {
22        if(a[i]>=0)
23            p++;
24    }
25    printf("\npositive numbers count%d",p);
26    for(i=0; i<6; i++)
27    {
28        if(a[i]<0)
29            n++;
30    }
31    printf("\nnegative numbers count%d",n);
32
33    return 0;
34 }
```

Output :

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

enter the list of integers : 10
-42
-8
13
11
-24

even numbers count4
odd numbers count2
positive numbers count3
negative numbers count3
PS C:\Users\user\OneDrive\Desktop\C Programming> █
Ln 5,
```

Q19- WAP to read a list of integers and store it in a 1D array. Write a C program to find the frequency of a particular number in a list of integers.

```
Terminal Help ← → C Programming
C book.c ● C pattern1.c C pattern2.c C file1.c C file2.c C file3.c C file4.c
C file5.c > main()
1 #include <stdio.h>
2
3 int main(){
4     int a[10],i,freq=0,key;
5     for(i=0; i<10; i++)
6     {
7         printf("\nEnter the list of integers :");
8         scanf("%d",&a[i]);
9     }
10    printf("\nEnter the integers to frequency");
11    scanf("%d",&key);
12    for(i=0; i<10; i++)
13    if(a[i]==key)
14        freq++;
15    printf("\nFrequency of a particular number in a list of integers is %d",key,freq);
16
17    return 0;
18 }
```

Output :

[PROBLEMS](#)[OUTPUT](#)[DEBUG CONSOLE](#)[TERMINAL](#)[PORTS](#)

```
enter the list of integers :2
enter the list of integers :7
enter the list of integers :8
enter the list of integers :13
enter the list of integers :45
enter the list of integers :31
enter the list of integers :56
enter the list of integers :1
enter the integers to frequency31
frequency of a particular number in a list of integers is 31
```

Q20- Develop a C function ISPRIME(num) that accepts an integer argument and returns 1 if the argument is prime, a 0 otherwise. Write a C program that invokes this function to generate prime numbers between the given ranges.

The screenshot shows a code editor interface with a dark theme. At the top, there's a navigation bar with tabs for 'Terminal', 'Help', and file icons. A search bar on the right contains the text 'C Programming'. Below the navigation bar, there are several tabs for different C files: 'pattern2.c', 'file1.c', 'file2.c', 'file3.c', 'file4.c', and 'file7.c' (which is the active tab). The main area displays the following C code:

```
file7.c > main()
1 #include <stdio.h>
2 int isprime(int num) {
3     if(num<=1) return 0;
4     for(int i=2; i*i<=num; i++) {
5         if(num%i==0) return 0;
6     }
7     return 1;
8 }
9 int main(){
10     int a,b;
11     printf("enter range : ");
12     scanf("%d%d",&a,&b);
13     printf("Prime numbers between %d and %d are:\n",a,b);
14     for(int i=a; i<=b; i++) {
15         if(isprime(i)) {
16             printf("%d\n",i);
17         }
18     }
19     printf("\n");
20     return 0;
21 }
```

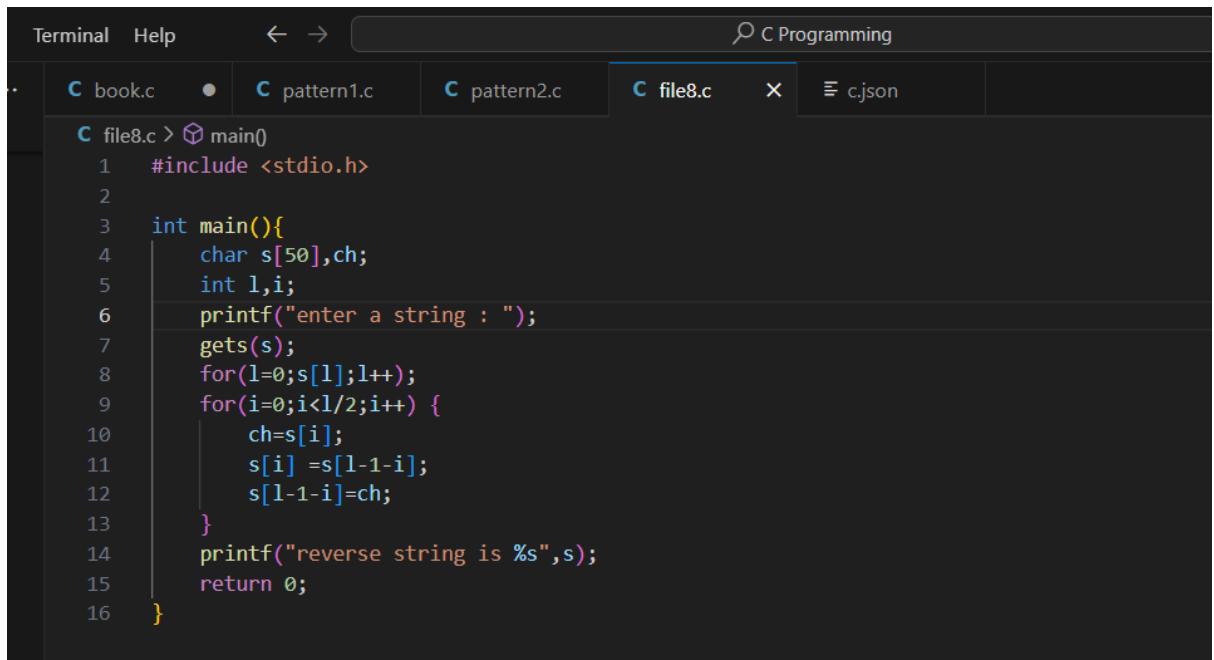
Output :

The screenshot shows a terminal window with a dark theme. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is the active tab), and 'PORTS'. The terminal window displays the following text:

```
enter range : 0
20
Prime numbers between 0 and 20 are:
2
3
5
7
11
13
17
19
```

At the bottom of the terminal window, it shows the path 'PS C:\Users\user\OneDrive\Desktop\C Programming>' followed by a cursor.

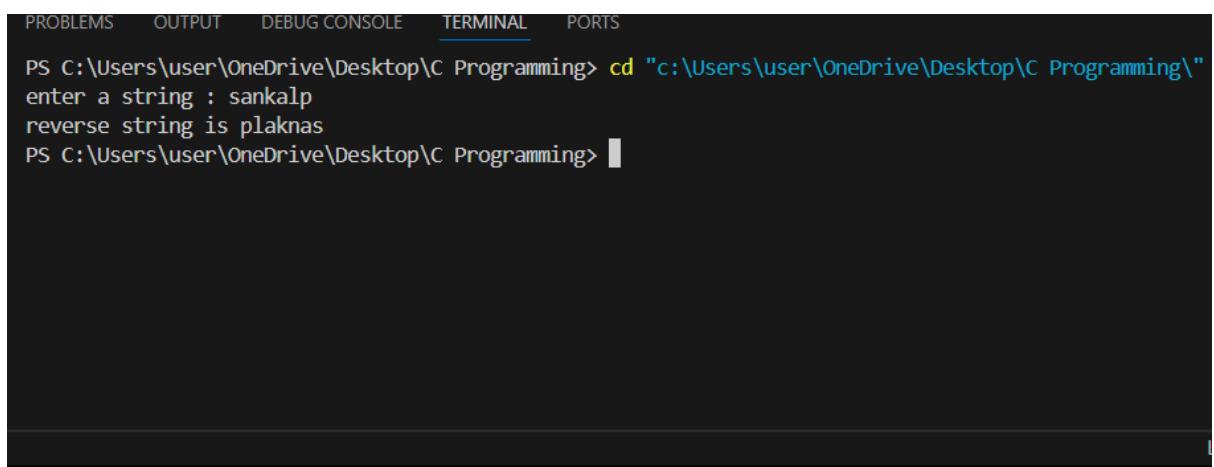
Q21-Develop a function REVERSE(str) that accepts a string argument. Write a C program that invokes this function to find the reverse of a given string.



The screenshot shows a code editor window titled "C Programming". The tabs at the top include "Terminal", "Help", and several file tabs: "book.c", "pattern1.c", "pattern2.c", "file8.c" (which is the active tab), and "c.json". The code in the editor is as follows:

```
C file8.c > main()
1 #include <stdio.h>
2
3 int main(){
4     char s[50],ch;
5     int l,i;
6     printf("enter a string : ");
7     gets(s);
8     for(l=0;s[l];l++);
9     for(i=0;i<l/2;i++) {
10         ch=s[i];
11         s[i] =s[l-1-i];
12         s[l-1-i]=ch;
13     }
14     printf("reverse string is %s",s);
15     return 0;
16 }
```

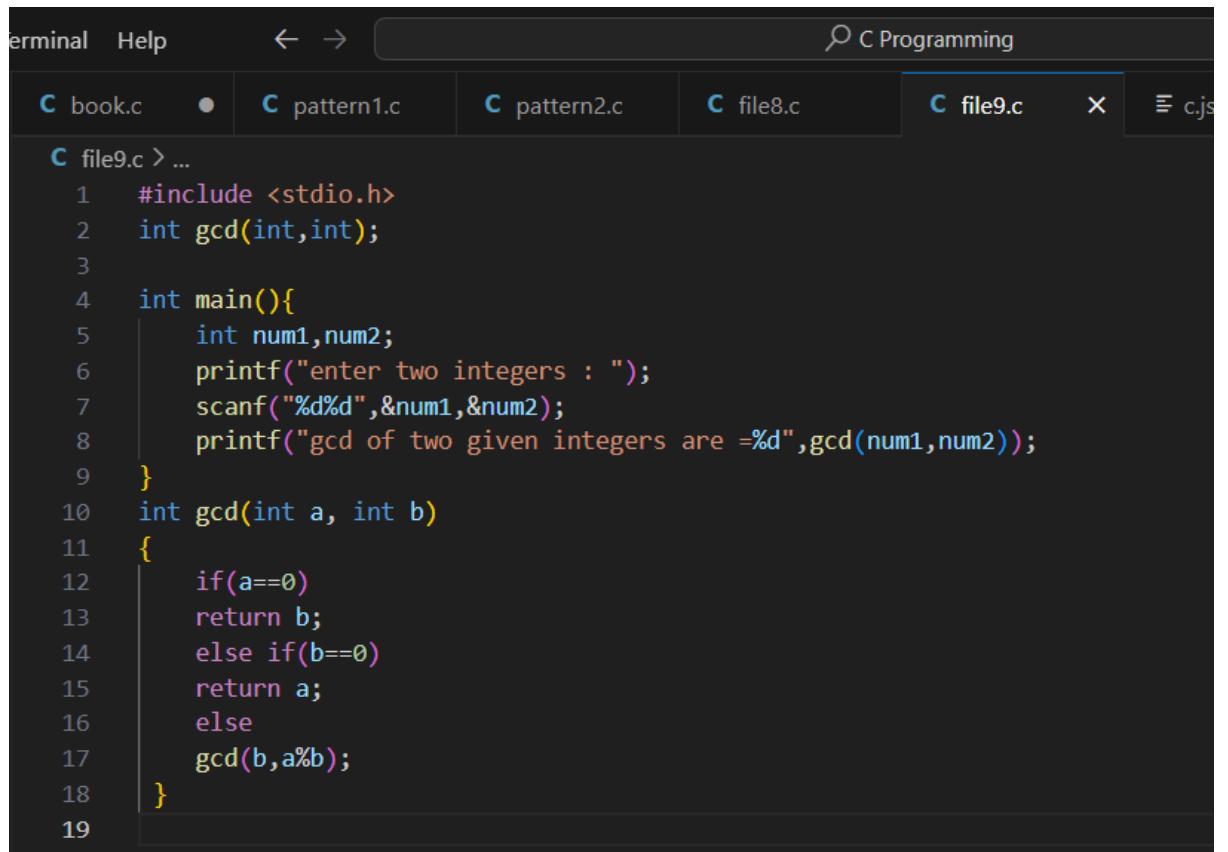
Output :



The screenshot shows a terminal window with tabs: "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL" (which is the active tab), and "PORTS". The terminal output is as follows:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\"
enter a string : sankalp
reverse string is plaknas
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

Q23-Develop a recursive function GCD(num1, num2) that accepts two integer arguments. Write a C program that invokes this function to find the greatest common divisor of two given integers.



The screenshot shows a code editor window titled "C Programming". The current file is "file9.c". The code implements a recursive function to calculate the GCD of two integers. It includes a main function that prompts the user for two integers and prints the result. The recursive function gcd handles the base cases where either integer is zero and returns the other integer, or both are non-zero, in which case it repeatedly calls itself with the second argument and the remainder of the first argument divided by the second.

```
C file9.c > ...
1 #include <stdio.h>
2 int gcd(int,int);
3
4 int main(){
5     int num1,num2;
6     printf("enter two integers : ");
7     scanf("%d%d",&num1,&num2);
8     printf("gcd of two given integers are =%d",gcd(num1,num2));
9 }
10 int gcd(int a, int b)
11 {
12     if(a==0)
13         return b;
14     else if(b==0)
15         return a;
16     else
17         gcd(b,a%b);
18 }
19
```

Output :

The screenshot shows a terminal window with the following text:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming"
enter two integers : 6
9
gcd of two given integers are =3
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

Q25-Develop a recursive function FIBO(num) that accepts an integer argument. Write a C program that invokes this function to generate the Fibonacci sequence up to num.

The screenshot shows a terminal window titled "C Programming". The current file is "file10.c". The code is as follows:

```
C file10.c > ...
1 #include <stdio.h>
2 int fibo(int num);
3
4 int main(){
5     int num, i;
6     printf("enter an integer : ");
7     scanf("%d",&num);
8     printf("fibonacci sequence up to %d is \n",num);
9     for(i=0; i<num; i++)
10    printf("%d",fibo(i));
11 }
12 int fibo(int num)
13 {
14     if(num==0)
15         return 0;
16     else if(num==1)
17         return 1;
18     else
19         return fibo(num-1)+fibo(num-2);
20 }
```

Output :

The screenshot shows a terminal window with the "TERMINAL" tab selected. The output is as follows:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) { gcc fi
enter an integer : 6
fibonacci sequence up to 6 is
011235
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

Ln 20, Col 5 Spaces: 4

Experiment- Structures

Q26- 1. Write a C program that uses functions to perform the following operations: a. Reading a complex number. b. Writing a complex number. c. Addition and subtraction of two complex numbers

```
Terminal Help ← → C Programming
C book.c ● C pattern1.c C pattern2.c C file8.c C file9.c C fil
C file11.c > ⌂ getcmp0
1 #include <stdio.h>
2 struct complex{
3     int real;
4     int imaginary;
5 };
6 void display(struct complex s) {
7     printf("%d+%d\n",s.real,s.imaginary);
8 }
9 struct complex getcmp(){
10    struct complex s;
11    printf("Enter real part : ");
12    scanf("%d",&s.real);
13    printf("Enter imaginary part : ");
14    scanf("%d",&s.imaginary);
15    return s;
16 }
17 struct complex sum(struct complex s1,struct complex s2) {
18     struct complex s;
19     s.real=s1.real+s2.real;
20     s.imaginary=s1.imaginary+s2.imaginary;
21     return s;
22 }
23 int main(){
24     struct complex s1,s2,s3;
25     s1=getcmp();
26     s2=getcmp();
27     display(s1);
28     display(s2);
29     s3=sum(s1,s2);
30     printf("The sum is : \n");
31     display(s3);
32     return 0;
33 }
```

Output :

```
18+59
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming\" ; if ($?) Enter real part : 12
Enter imaginary part : 55
Enter real part : 5
Enter imaginary part : 9
12+55
5+9
The sum is :
17+64
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

Ln 14, Col 17

Q27- Write a C program to compute the monthly pay of 100 employees using each employee's name, basic pay. The DA is computed as 52% of the basic pay. Gross-salary (basic pay + DA). Print the employees name and gross salary.

The screenshot shows a code editor interface for C Programming. The file being edited is Structure2.c. The code defines a struct employee with fields name (char array), bp (float), and gs (float). It then initializes an array of 100 employees and loops through the first three to input name and salary, calculate gross salary (gs = bp + 0.52 * bp), and print the results.

```
Structure2.c > main()
1 #include <stdio.h>
2 struct employee{
3     char name[30];
4     float bp,gs;
5 };
6 int main()
7 {
8     struct employee e[100];
9     for(int i=1; i<=3; i++)
10    {
11        float da,gs;
12        printf("Enter employee details :\n");
13        printf("Name :");
14        scanf("%s",e[i].name);
15        printf("Salary :");
16        scanf("%f",&e[i].bp);
17        da=0.52*e[i].bp;
18        e[i].gs=e[i].bp+da;
19
20        printf("Gross salary : %f\n",e[i].gs);
21    }
22    printf("Employee details are : \n");
23    for(int i=1; i<=3; i++)
24    {
25        printf("Name : %s, Salary : %f, Gross salary : %f\n",e[i].name,e[i].bp,e[i].gs);
26    }
27 }
```

Output :

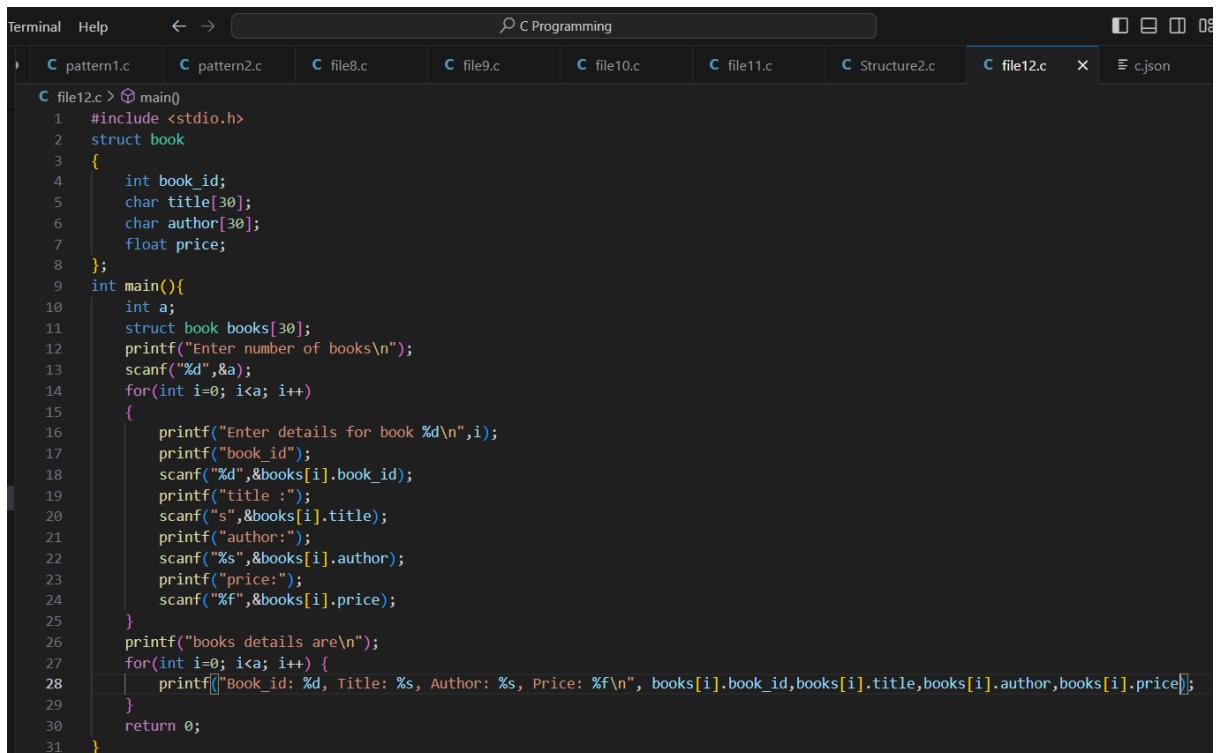
The terminal window shows the execution of the program. It prompts for employee details, including name and salary, and then prints the name, salary, and gross salary for each employee.

```
27 }
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

re2 }
Enter employee details :
Name :
rohit
Salary :500000
Gross salary : 760000.000000
Enter employee details :
Name :

```

Q28-Create a Book structure containing book_id, title, author name and price. Write a C program to pass a structure as a function argument and print the book details



```
Terminal Help ← → C Programming
c pattern1.c c pattern2.c c file8.c c file9.c c file10.c c file11.c c Structure2.c c file12.c x c.json
file12.c > ↻ main()
1 #include <stdio.h>
2 struct book
3 {
4     int book_id;
5     char title[30];
6     char author[30];
7     float price;
8 };
9 int main(){
10     int a;
11     struct book books[30];
12     printf("Enter number of books\n");
13     scanf("%d",&a);
14     for(int i=0; i<a; i++)
15     {
16         printf("Enter details for book %d\n",i);
17         printf("book_id");
18         scanf("%d",&books[i].book_id);
19         printf("title :");
20         scanf("s",&books[i].title);
21         printf("author:");
22         scanf("%s",&books[i].author);
23         printf("price:");
24         scanf("%f",&books[i].price);
25     }
26     printf("books details are\n");
27     for(int i=0; i<a; i++) {
28         printf("Book_id: %d, Title: %s, Author: %s, Price: %f\n", books[i].book_id,books[i].title,books[i].author,books[i].price);
29     }
30     return 0;
31 }
```

Q29- Create a union containing 6 strings: name, home_address, hostel_address, city, state and zip. Write a C program to display your present address.

terminal Help ← → C Programming

C pattern2.c	C file8.c	C file9.c	C file10.c	C file11.c
--------------	-----------	-----------	------------	------------

```
C file13.c > main()
1 #include <stdio.h>
2 #include <string.h>
3
4 union details {
5     char name[20];
6     char home_address[30];
7     char hostel_address[30];
8     char city[10];
9     char state[10];
10    int zip;
11 }a;
12 int main(){
13     strcpy(a.name, "rohit");
14     printf("\n%s", a.name);
15
16     strcpy(a.home_address, "71- Rajpur Road");
17     printf("\n%s", a.home_address);
18
19     strcpy(a.hostel_address, "69- Stanza Living");
20     printf("\n%s", a.hostel_address);
21
22     strcpy(a.city, "Dehradun");
23     printf("\n%s", a.city);
24
25     strcpy(a.state, "Uttarakhand");
26     printf("\n%s", a.state);
27
28     a.zip = 248001;
29     printf("\n%d", a.zip);
30     return 0;
31 }
```

Output :

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is underlined), and 'PORTS'. The terminal window displays the following text:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming"
rohit
71- Rajpur Road
69- Stanza Living
71- Rajpur Road
69- Stanza Living
69- Stanza Living
Dehradun
Uttarakhand
248001
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

30) Write a program to define some constant variable in preprocessor.

Input:

A screenshot of a code editor window titled "C Programming". The tabs at the top show "Terminal", "Help", and several C files: "DMA.c", "union4.c", "union5.c", "constantvar.c" (which is the active tab), and "dire...". The code in "constantvar.c" is as follows:

```
C constantvar.c > ⚙ main()
1 #include <stdio.h>
2
3 #define PI 3.14159
4 #define MAX_LENGTH 100
5 #define GREETING "Hello, World!"
6
7 int main() {
8
9     printf("The value of PI is: %.5f\n", PI);
10    printf("The maximum length is: %d\n", MAX_LENGTH);
11    printf("%s\n", GREETING);
12
13    return 0;
14 }
15
```

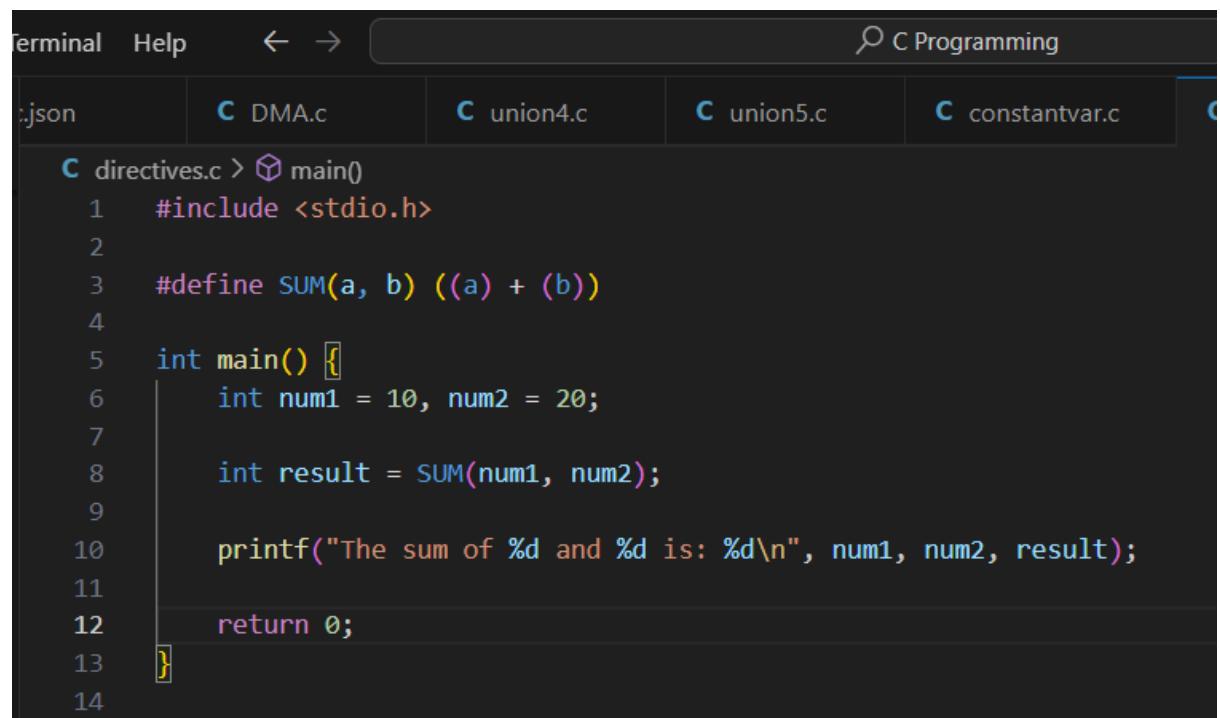
Output :

A screenshot of a terminal window titled "C Programming". The tabs at the top show "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL" (which is the active tab), and "PORTS". The terminal output shows the execution of the program "constantvar.c":

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming"
PS C:\Users\user\OneDrive\Desktop\C Programming> antvar }
The value of PI is: 3.14159
The maximum length is: 100
Hello, World!
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

32) Write a program to define a function in directives.

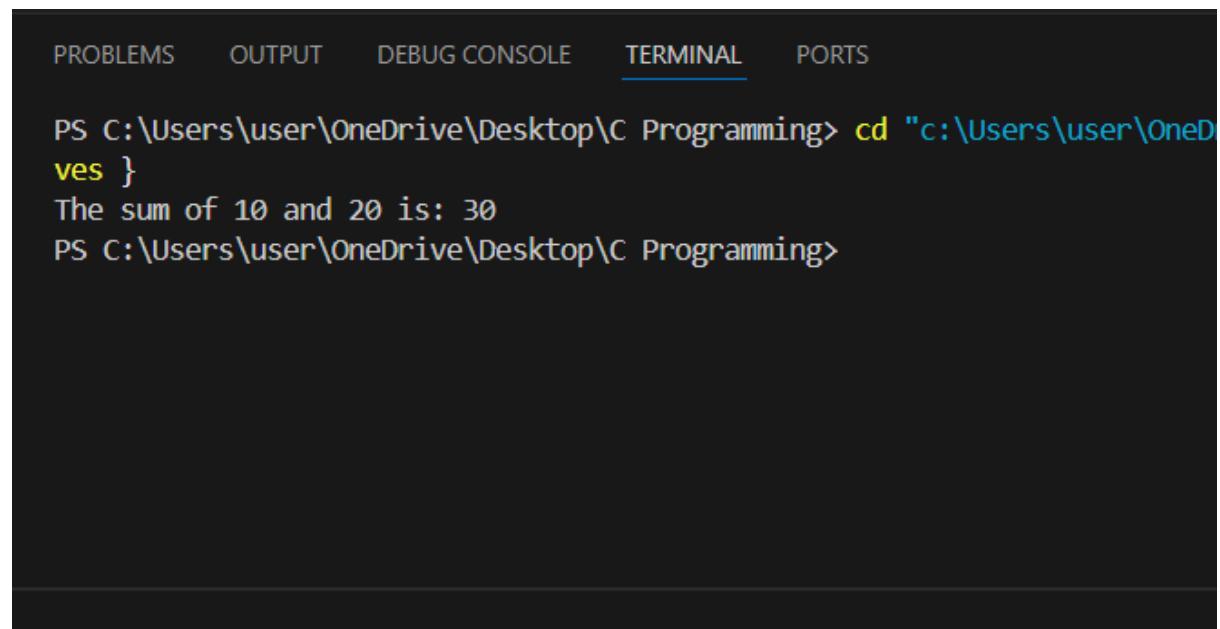
Input :



The screenshot shows a terminal window with a dark theme. At the top, there are tabs for "Terminal", "Help", and a search bar containing "C Programming". Below the tabs, there are several file icons: ".json", "DMA.c", "union4.c", "union5.c", and "constantvar.c". The main area of the terminal displays the following C code:

```
C directives.c > ⚙ main()
1 #include <stdio.h>
2
3 #define SUM(a, b) ((a) + (b))
4
5 int main()
6 {
7     int num1 = 10, num2 = 20;
8
9     int result = SUM(num1, num2);
10
11    printf("The sum of %d and %d is: %d\n", num1, num2, result);
12
13    return 0;
14 }
```

Output :

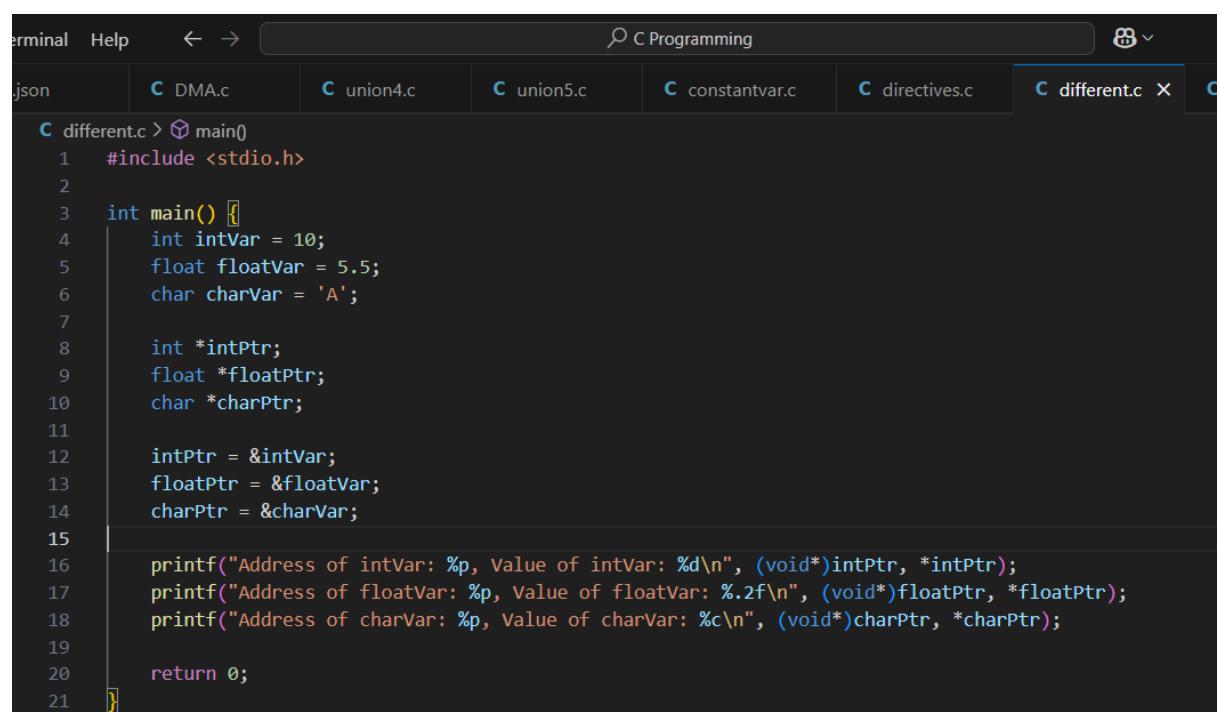


The screenshot shows a terminal window with a dark theme. At the top, there are tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", and "PORTS", with "TERMINAL" being underlined. The main area of the terminal displays the following text:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming"
The sum of 10 and 20 is: 30
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

33) Declare different types of pointers (int, float, char) and initialize them with the addresses of variables. Print the values of both the pointers and the variables they point to

Input :



The screenshot shows a code editor window with a dark theme. The title bar says "C Programming". The tabs at the top include "json", "DMA.c", "union4.c", "union5.c", "constantvar.c", "directives.c", and "different.c" (which is the active tab). The code in the editor is as follows:

```
C different.c > main()
1 #include <stdio.h>
2
3 int main() {
4     int intVar = 10;
5     float floatVar = 5.5;
6     char charVar = 'A';
7
8     int *IntPtr;
9     float *floatPtr;
10    char *charPtr;
11
12    IntPtr = &intVar;
13    floatPtr = &floatVar;
14    charPtr = &charVar;
15
16    printf("Address of intVar: %p, Value of intVar: %d\n", (void*)IntPtr, *IntPtr);
17    printf("Address of floatVar: %p, Value of floatVar: %.2f\n", (void*)floatPtr, *floatPtr);
18    printf("Address of charVar: %p, Value of charVar: %c\n", (void*)charPtr, *charPtr);
19
20    return 0;
21 }
```

Output :

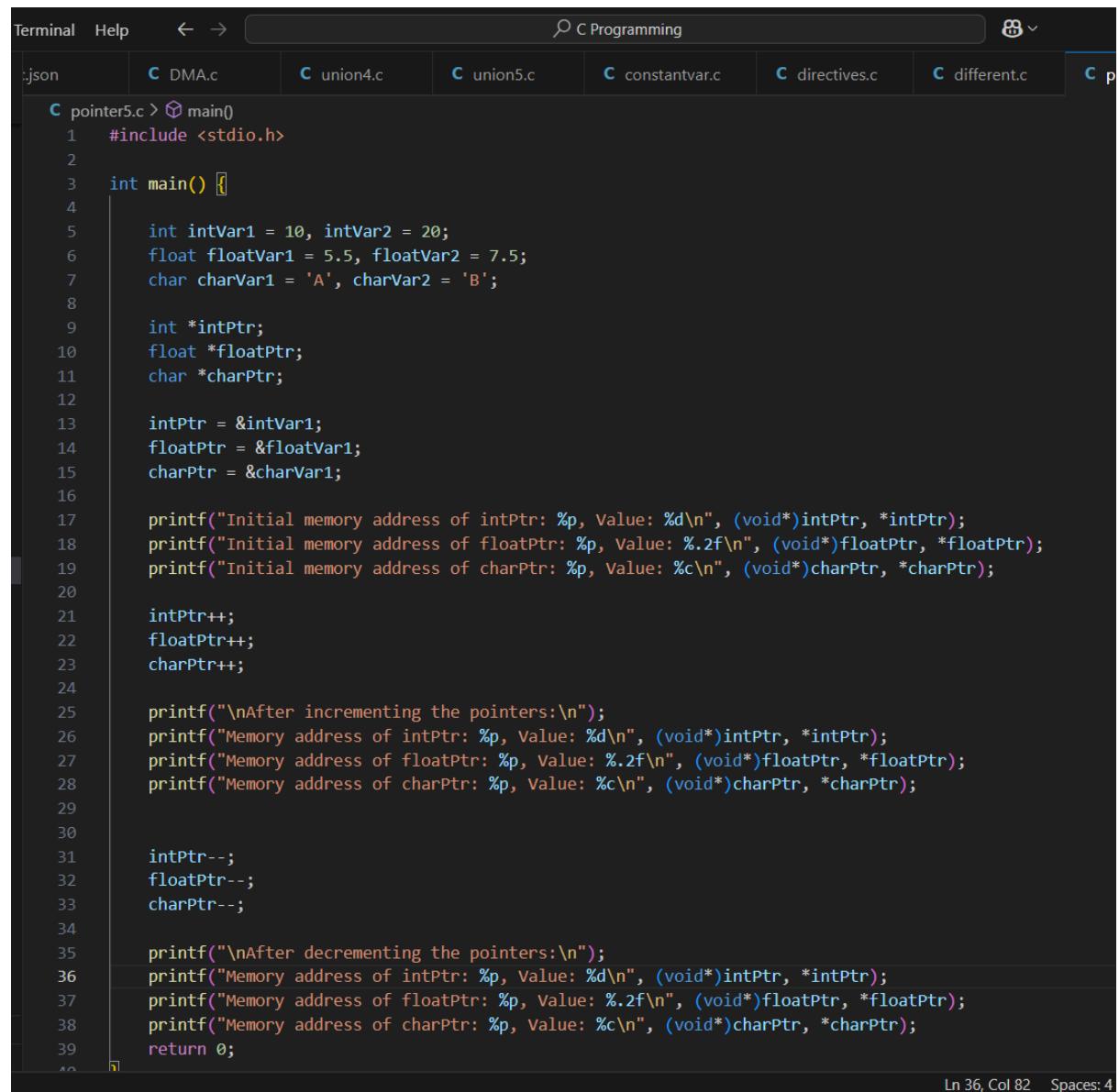


The screenshot shows a terminal window with a dark theme. The tabs at the top are "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL" (which is the active tab), and "PORTS". The terminal output is as follows:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Programming"
}
Address of intVar: 0061FF10, Value of intVar: 10
Address of floatVar: 0061FF0C, Value of floatVar: 5.50
Address of charVar: 0061FF0B, Value of charVar: A
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

34) Perform pointer arithmetic (increment and decrement) on pointers of different data types. Observe how the memory addresses change and the effects on data access.

Input :



The screenshot shows a terminal window with a dark theme. The title bar says "C Programming". The tab bar includes "Terminal", "Help", and several C files: "DMA.c", "union4.c", "union5.c", "constantvar.c", "directives.c", "different.c", and the current file "pointer5.c". The code in "pointer5.c" demonstrates pointer arithmetic on integers, floats, and characters. It prints initial memory addresses, increments or decrements the pointers, and then prints the final memory addresses. The code is as follows:

```
1 #include <stdio.h>
2
3 int main() {
4
5     int intVar1 = 10, intVar2 = 20;
6     float floatVar1 = 5.5, floatVar2 = 7.5;
7     char charVar1 = 'A', charVar2 = 'B';
8
9     int *intPtr;
10    float *floatPtr;
11    char *charPtr;
12
13    intPtr = &intVar1;
14    floatPtr = &floatVar1;
15    charPtr = &charVar1;
16
17    printf("Initial memory address of intPtr: %p, Value: %d\n", (void*)intPtr, *intPtr);
18    printf("Initial memory address of floatPtr: %p, Value: %.2f\n", (void*)floatPtr, *floatPtr);
19    printf("Initial memory address of charPtr: %p, Value: %c\n", (void*)charPtr, *charPtr);
20
21    intPtr++;
22    floatPtr++;
23    charPtr++;
24
25    printf("\nAfter incrementing the pointers:\n");
26    printf("Memory address of intPtr: %p, Value: %d\n", (void*)intPtr, *intPtr);
27    printf("Memory address of floatPtr: %p, Value: %.2f\n", (void*)floatPtr, *floatPtr);
28    printf("Memory address of charPtr: %p, Value: %c\n", (void*)charPtr, *charPtr);
29
30
31    intPtr--;
32    floatPtr--;
33    charPtr--;
34
35    printf("\nAfter decrementing the pointers:\n");
36    printf("Memory address of intPtr: %p, Value: %d\n", (void*)intPtr, *intPtr);
37    printf("Memory address of floatPtr: %p, Value: %.2f\n", (void*)floatPtr, *floatPtr);
38    printf("Memory address of charPtr: %p, Value: %c\n", (void*)charPtr, *charPtr);
39    return 0;
40 }
```

Ln 36, Col 82 Spaces: 4

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Initial memory address of charPtr: 0061FEFF, Value: A

After incrementing the pointers:
Memory address of intPtr: 0061FF08, Value: 6422272
Memory address of floatPtr: 0061FF04, Value: 0.00
Memory address of charPtr: 0061FF00, Value:

After decrementing the pointers:
Memory address of intPtr: 0061FF04, Value: 10
Memory address of floatPtr: 0061FF00, Value: 5.50
Memory address of charPtr: 0061FEFF, Value: A
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

35) Write a function that accepts pointers as parameters. Pass variables by reference using pointers and modify their values within the function.

Input :

The screenshot shows a code editor interface with a dark theme. At the top, there's a navigation bar with 'Terminal' and 'Help' on the left, and a search bar with 'C Programming' on the right. Below the search bar is a tab bar with several tabs, some of which are partially visible. The main area contains the source code for 'parameter.c'. The code defines a function 'modifyValues' that takes three pointers as arguments and assigns them specific values. It then prints the initial values of 'intVar', 'floatVar', and 'charVar', calls 'modifyValues', and prints the modified values.

```
C parameter.c > main()
1 #include <stdio.h>
2
3 void modifyValues(int *intPtr, float *floatPtr, char *charPtr) {
4
5     *intPtr = 100;
6     *floatPtr = 3.14f;
7     *charPtr = 'Z';
8 }
9
10 int main() {
11     int intVar = 10;
12     float floatVar = 5.5;
13     char charVar = 'A';
14
15     printf("Before modification:\n");
16     printf("intVar: %d, floatVar: %.2f, charVar: %c\n", intVar, floatVar, charVar);
17
18     modifyValues(&intVar, &floatVar, &charVar);
19
20     printf("\nAfter modification:\n");
21     printf("intVar: %d, floatVar: %.2f, charVar: %c\n", intVar, floatVar, charVar);
22
23     return 0;
24 }
```

Output :

The screenshot shows a terminal window with a dark theme. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is underlined, indicating it's the active tab), and 'PORTS'. The terminal output shows the command 'cd' being used to change the directory to 'C Programming'. It then displays the output of the program, which includes the initial values of 'intVar', 'floatVar', and 'charVar', followed by the modified values after the 'modifyValues' function is called.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

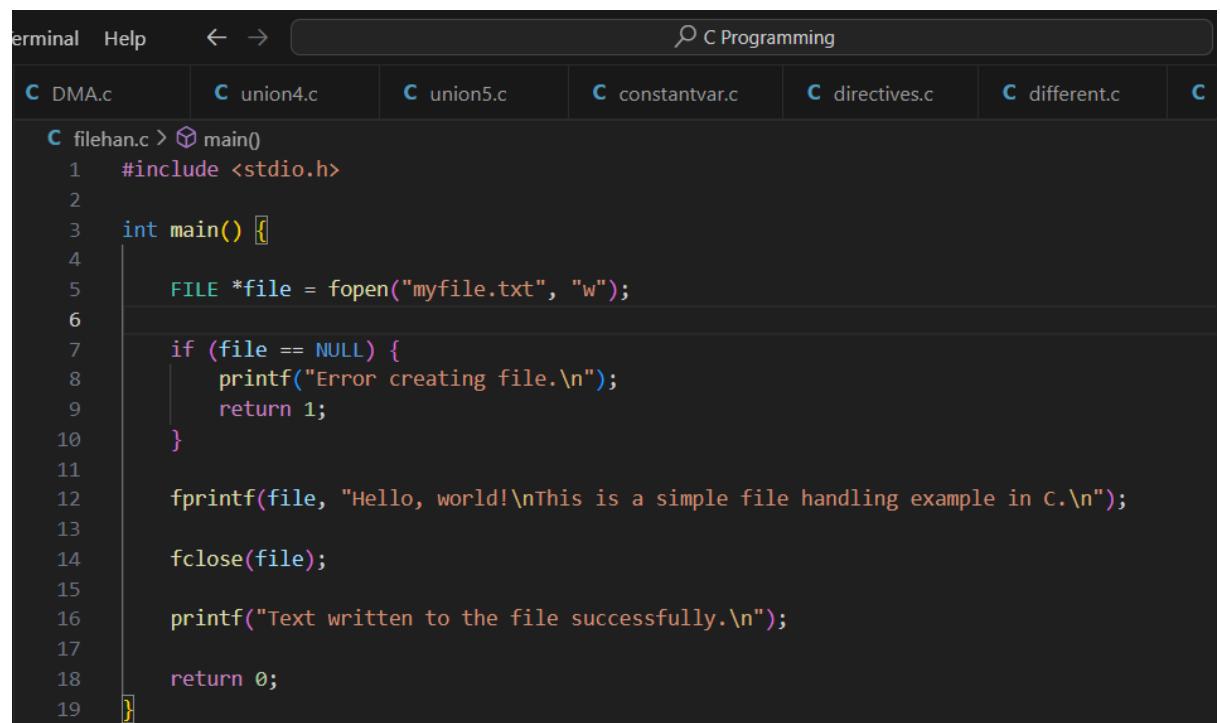
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C Pr
    }
Before modification:
intVar: 10, floatVar: 5.50, charVar: A

After modification:
intVar: 100, floatVar: 3.14, charVar: Z
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

36) File handling in C

1) Write a program to create a new file and write text into it.

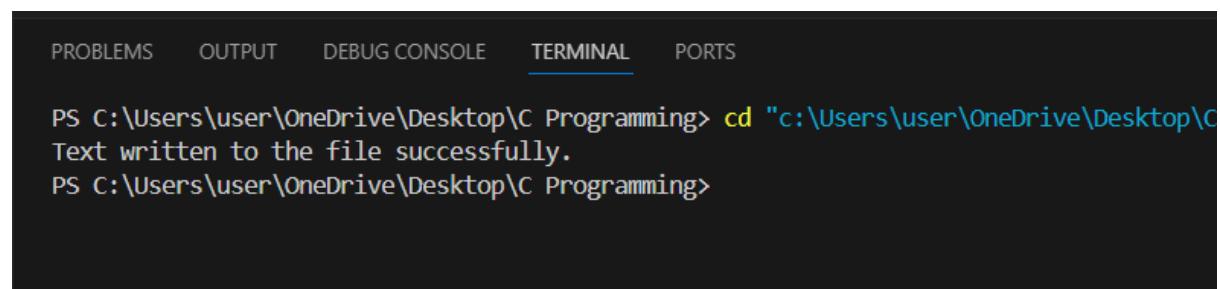
Input :



A screenshot of a terminal window titled "C Programming". The window shows a list of files at the top: DMA.c, union4.c, union5.c, constantvar.c, directives.c, different.c, and filehan.c. The file "filehan.c" is open and displayed in the main area. The code is as follows:

```
C filehan.c > ⇧ main()
1 #include <stdio.h>
2
3 int main() {
4
5     FILE *file = fopen("myfile.txt", "w");
6
7     if (file == NULL) {
8         printf("Error creating file.\n");
9         return 1;
10    }
11
12    fprintf(file, "Hello, world!\nThis is a simple file handling example in C.\n");
13
14    fclose(file);
15
16    printf("Text written to the file successfully.\n");
17
18    return 0;
19 }
```

Output :

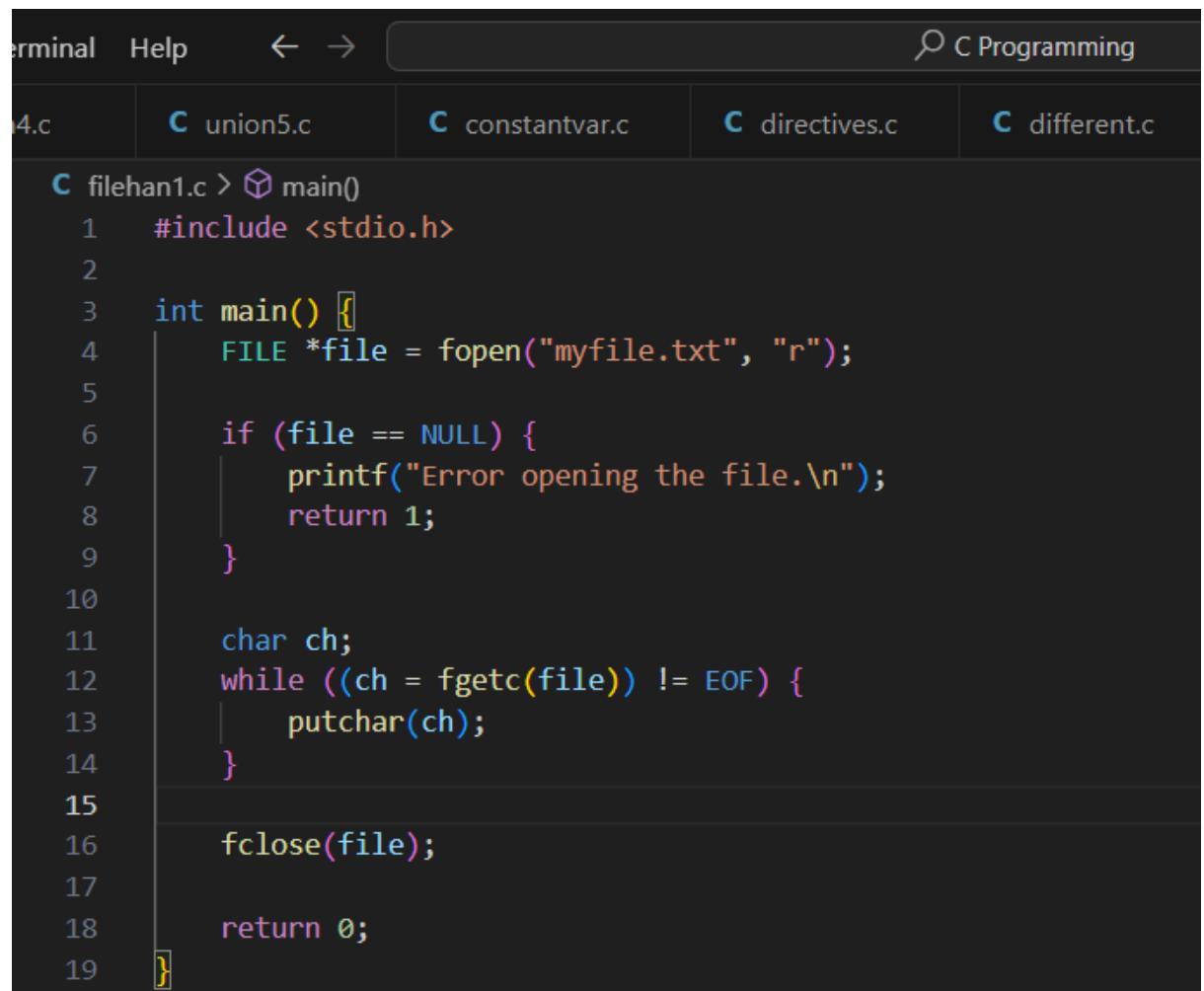


A screenshot of a terminal window showing the output of the C program. The terminal tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The terminal output is as follows:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C"
Text written to the file successfully.
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

2) Open an existing file and read its content character by character, and then close the file.

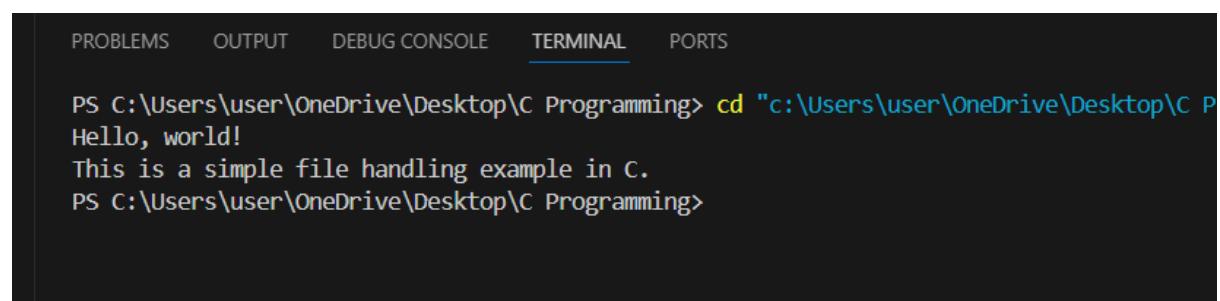
Input :



The screenshot shows a code editor interface with a dark theme. At the top, there's a navigation bar with 'Terminal' and 'Help' on the left, and a search bar with 'C Programming' on the right. Below the navigation bar, there are tabs for several files: '4.c', 'union5.c', 'constantvar.c', 'directives.c', and 'different.c'. The current file is 'filehan1.c'. The code in 'filehan1.c' is as follows:

```
C filehan1.c > main()
1 #include <stdio.h>
2
3 int main() {
4     FILE *file = fopen("myfile.txt", "r");
5
6     if (file == NULL) {
7         printf("Error opening the file.\n");
8         return 1;
9     }
10
11    char ch;
12    while ((ch = fgetc(file)) != EOF) {
13        putchar(ch);
14    }
15
16    fclose(file);
17
18    return 0;
19 }
```

Output :

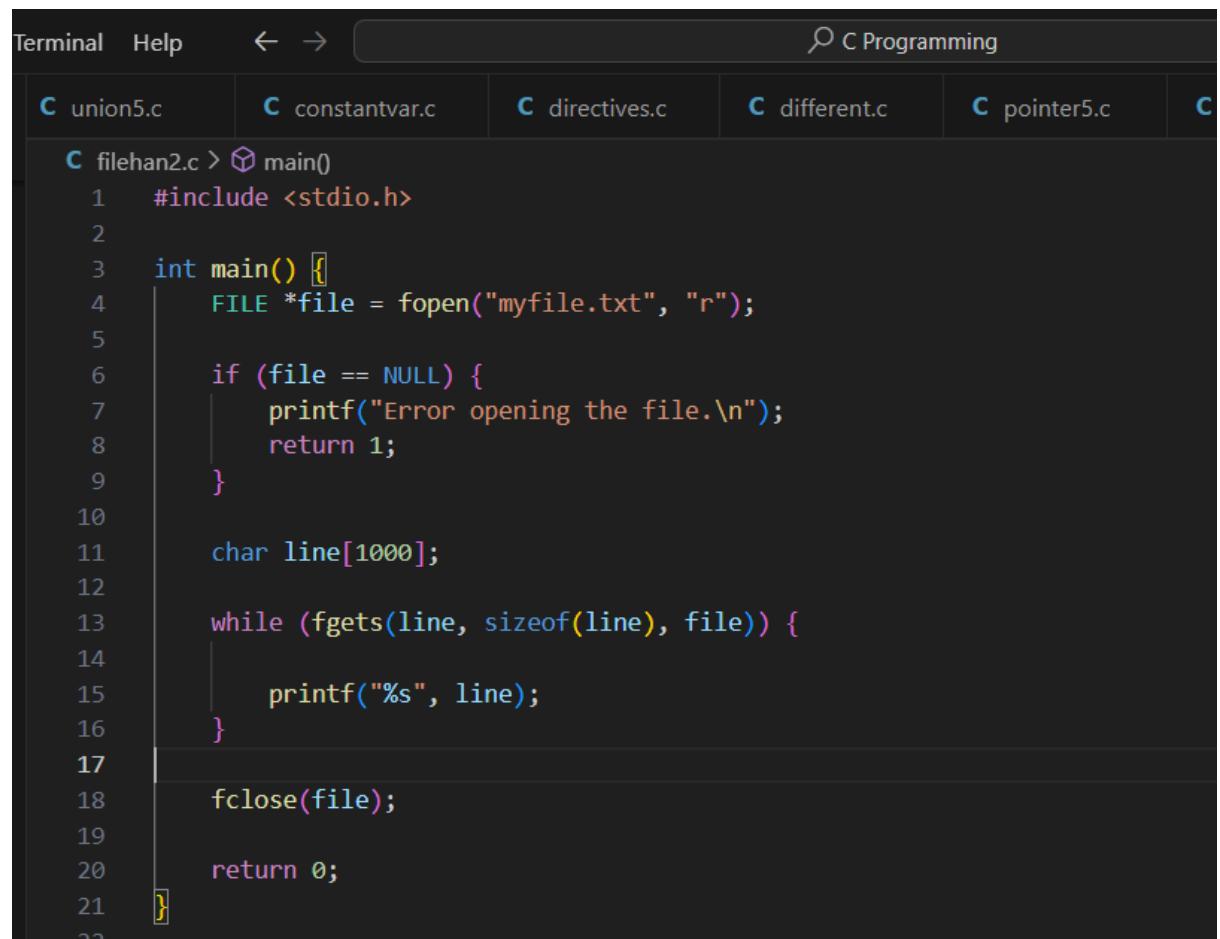


The screenshot shows a terminal window with a dark theme. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is underlined, indicating it's active), and 'PORTS'. The terminal output is as follows:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\C P
Hello, world!
This is a simple file handling example in C.
PS C:\Users\user\OneDrive\Desktop\C Programming>
```

3) Open a file, read its content line by line, and display each line on the console.

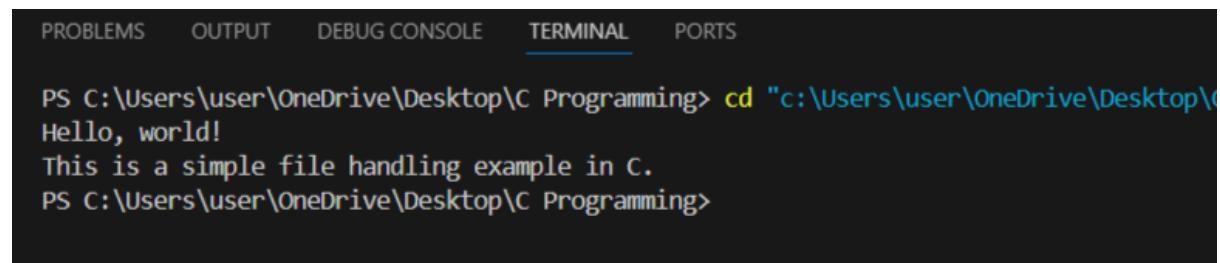
Input :



The screenshot shows a code editor interface with a dark theme. At the top, there's a navigation bar with tabs for "Terminal", "Help", and file icons. To the right of the tabs is a search bar containing "C Programming". Below the navigation bar, there are several tabs for other C files: "union5.c", "constantvar.c", "directives.c", "different.c", "pointer5.c", and "filehan2.c". The "filehan2.c" tab is currently active, indicated by a blue border. The code editor displays the following C program:

```
1 #include <stdio.h>
2
3 int main() {
4     FILE *file = fopen("myfile.txt", "r");
5
6     if (file == NULL) {
7         printf("Error opening the file.\n");
8         return 1;
9     }
10
11    char line[1000];
12
13    while (fgets(line, sizeof(line), file)) {
14
15        printf("%s", line);
16    }
17
18    fclose(file);
19
20    return 0;
21 }
22
```

Output :



The screenshot shows a terminal window with a dark theme. At the top, there are tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", and "PORTS". The "TERMINAL" tab is currently selected, indicated by an underline. The terminal window displays the following text:

```
PS C:\Users\user\OneDrive\Desktop\C Programming> cd "c:\Users\user\OneDrive\Desktop\"
Hello, world!
This is a simple file handling example in C.
PS C:\Users\user\OneDrive\Desktop\C Programming>
```