Name	Vedans Rawat	Year	2025
Subject		Class	Batch 20
Semester	First	Roll No.	590025450

INDEX

Sr. No.	Experiment Description	Experiment Date	Submission Date	Remarks / Signature
1.	Experiment-1-Installation, Environment setup & starting with C language. • WAP to print "Hello World" • WAP to print address in multiple line. • WAP to prompt user to enter their name & age. • WAP to add two numbers, take no. from user.	13/08/2025	20/08/2025	
2.	Experiment-2-Operator • WAP to calculate the area & perimeter of rectangle based on its length & width. • WAP to convert temperature from Celcius to Farenheit.	13/08/2025	20/08/2025	
3.	Experiment-11-Bitwise Operator • WAP to apply bitwise OR, AND, NOT operators • WAP to apply left shift & right shift operator.	20/08/2025	20/08/2025	

INDEX

Sr. No.	Experiment Description	Experiment Date	Submission Date	Remarks / Signature
4.	Experiment-3-Conditional Statements •WAP to check if the triangle is valid or not. If the validity is established, do check if the triangle is isosceles, right angle, or scalene. Take sides of the triangle as input from a user. •WAP to compute the BMI of a person & print BMI value as per the following ranges.	20/08/2025	20/08/2025	
5.	Experiment-4	14/09/2025	29/10/2025	
6.	Experiment-5	17/09/2025	29/10/2025	
7.	Experiment-6	30/09/2025	29/10/2025	
8.	Experiment-8	30/09/2025	29/10/2025	
9.	Experiment-7			
10.	Experiment-9			
11.	Experiment-10			
12.	Experiment-12			
13.	Experiment-13			
14.	Experiment-14			
15.	Experiment-15			

Date

C - PROGRAMING

LAB FILE

Mohan

CLASSMATE
Date :
Page :

EXPERIMENT 1: INSTALLATION, ENVIRONMENT SETUP & STARTING WITH C LANGUAGE

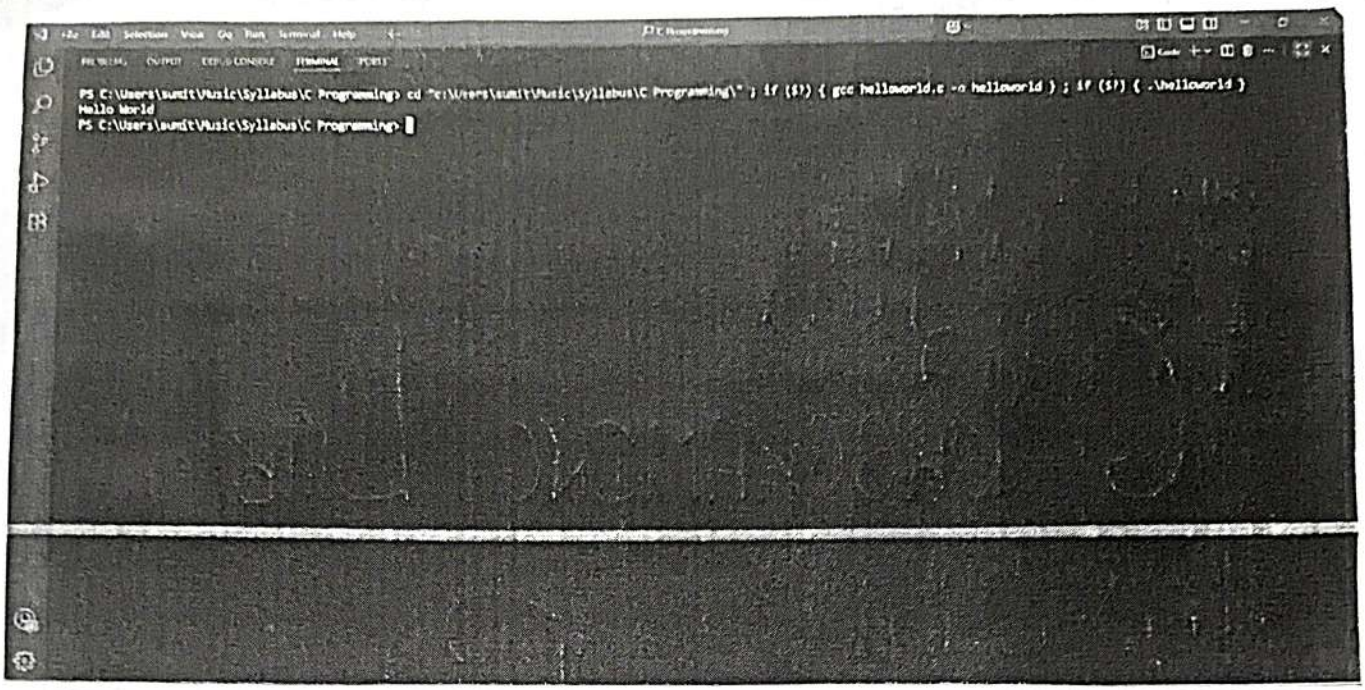
1. Write a C program to print "Hello World".

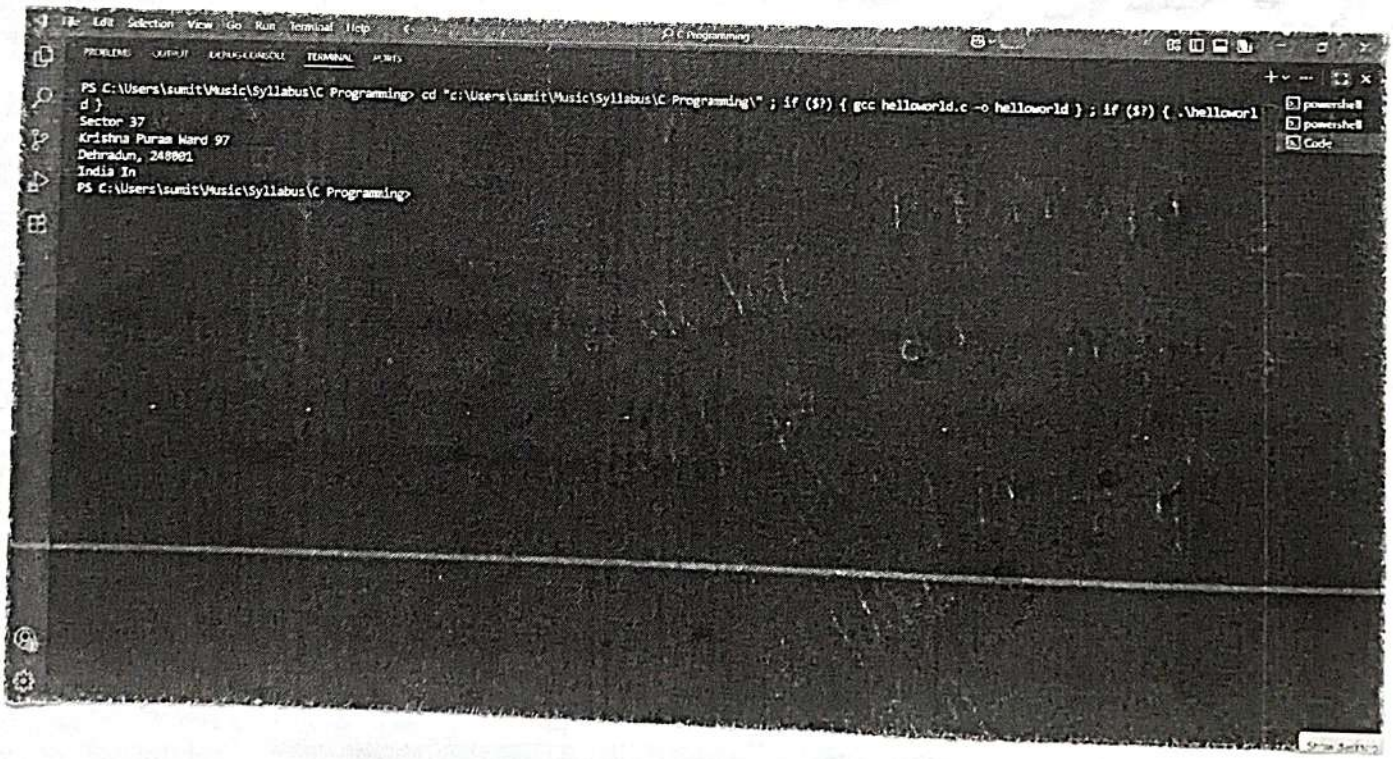
Prog:
```
#include <stdio.h>

int main() {
    printf("Hello World\n");
    return 0;
}
```

2. Write a C program to print the address in multiple line (new lines).

Prog:
```
#include <stdio.h>
int main() {
    printf("Sector 37\n");
    printf("Krishna Puram Ward 37\n");
    printf("Dehradun, 248001\n");
    printf("India\n");
    return 0;
}
```


PS C:\Users\sumit\Music\Syllabus\C Programming> cd "c:\Users\sumit\Music\Syllabus\C Programming\" ; if ($?) { gcc helloworld.c -o helloworld } ; if ($?) { .\helloworld }
Hello World
PS C:\Users\sumit\Music\Syllabus\C Programming>


PS C:\Users\sumit\Music\Syllabus\C Programming> cd "c:\Users\sumit\Music\Syllabus\C Programming\" ; if ($?) { gcc helloworld.c -o helloworld } ; if ($?) { .\helloworl
d }
Sector 37
Krishna Puram Ward 97
Dehradun, 248001
India In
PS C:\Users\sumit\Music\Syllabus\C Programming>

CLASSMATE
Date :
Page :

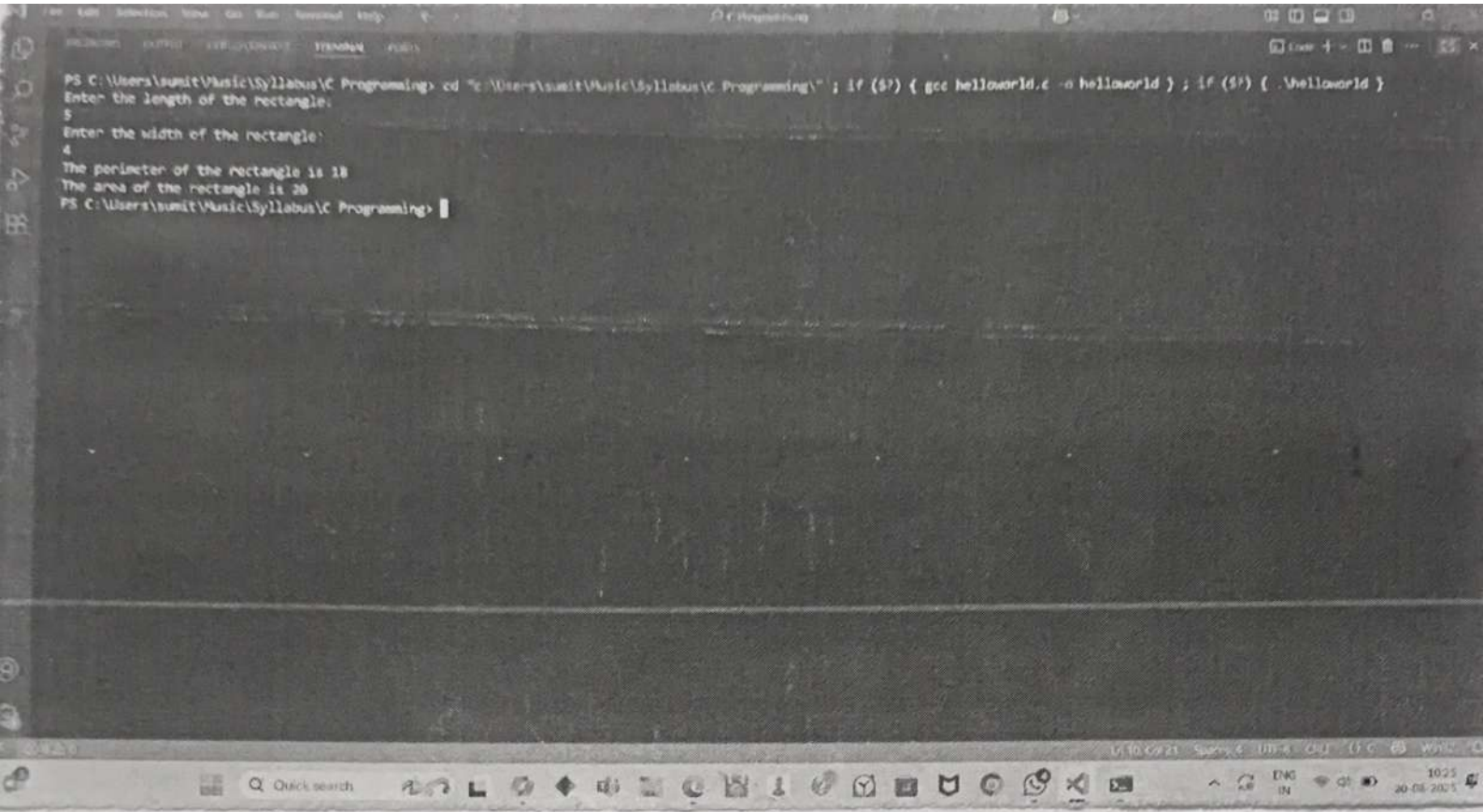
EXPERIMENT 2- OPERATOR

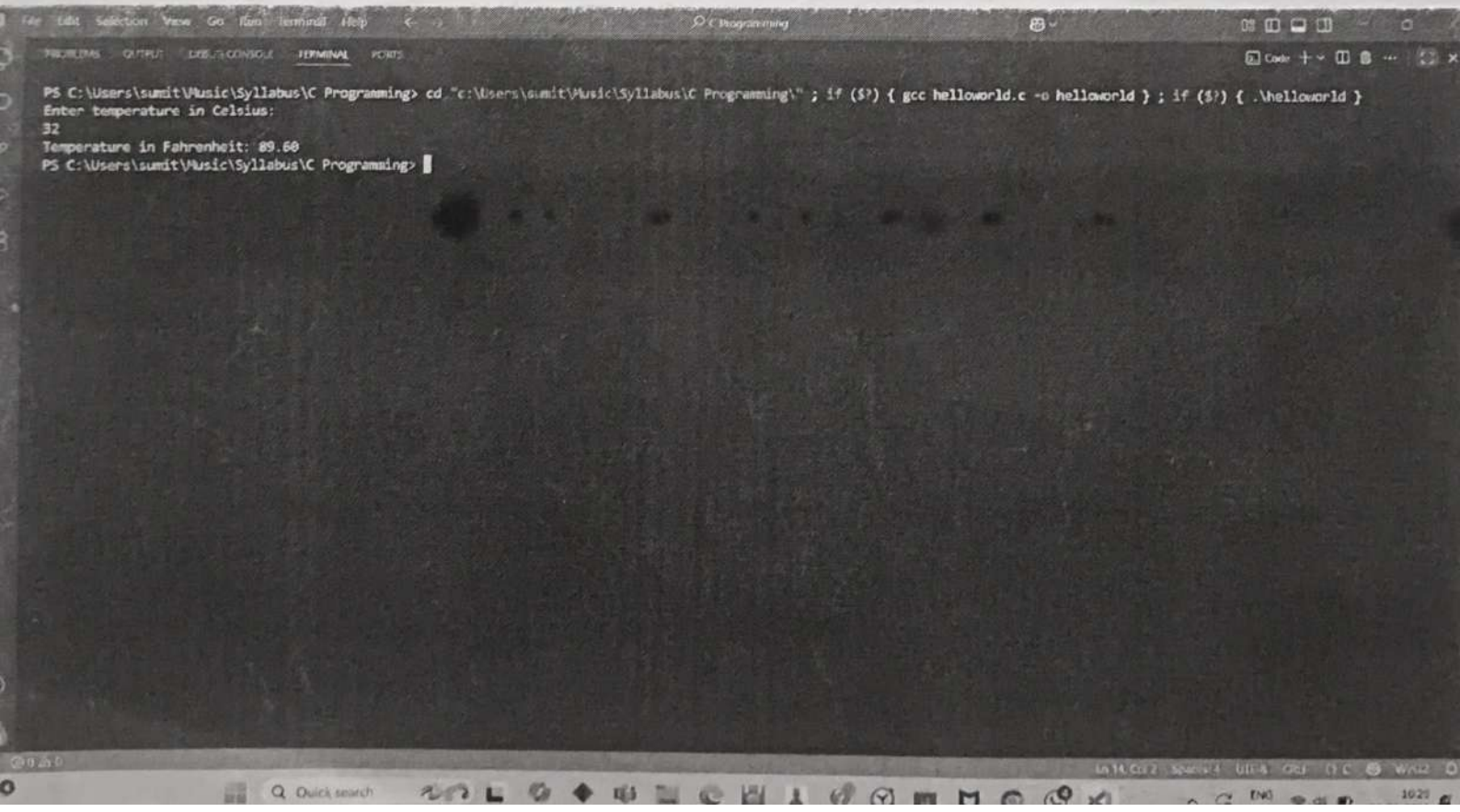
1. Write a C program to calculate the area & perimeter of rectangle based on its length & width.

```
#include <stdio.h>
    float main(){
    float length, width, area, peri;
        printf("Enter the length: \n");
        scanf("%f", &length);
        printf("Enter the width" \n");
        scanf("%f", &width);
        area = length * width;
        peri = 2*(length + width);
        printf("The area of rectangle is = %f \n", area);
        printf("The perimeter of rectangle is = %f \n", peri);
        return 0;
}
```

2. Write a C program to convert temperature from Celcius to Farenheit using the formula F = (C * 9/5) + 32.

```
#include <stdio.h>
    float main(){
        printf("Enter temperature in Celcius: \n");
        scanf("%f", Celcius);
        f = (Celcius * 9/5) + 32;
        printf("Given temperature in farenheit is = %f \n", f);
        return 0;
}
```


PS C:\Users\sumit\Music\Syllabus\C Programming> cd "c:\Users\sumit\Music\Syllabus\C Programming\" ; if ($?) { gcc helloworld.c -o helloworld } ; if ($?) { .\helloworld }
Enter the length of the rectangle:
5
Enter the width of the rectangle:
4
The perimeter of the rectangle is 18
The area of the rectangle is 20
PS C:\Users\sumit\Music\Syllabus\C Programming>


PS C:\Users\sumit\Music\Syllabus\C Programming> cd "c:\Users\sumit\Music\Syllabus\C Programming\" ; if ($?) { gcc helloworld.c -o helloworld } ; if ($?) { .\helloworld }
Enter temperature in Celsius:
32
Temperature in Fahrenheit: 89.60
PS C:\Users\sumit\Music\Syllabus\C Programming>

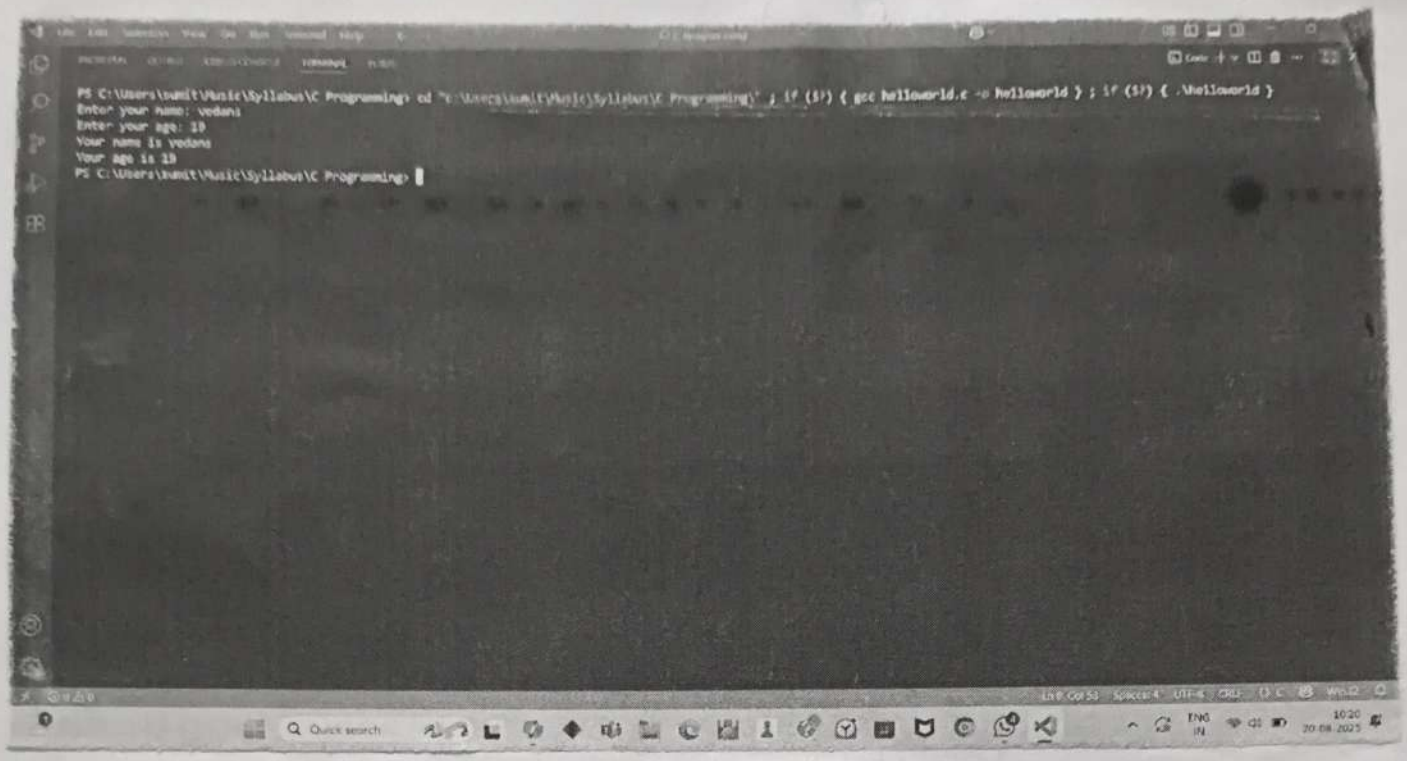
3. Write a program that prompts the user to enter their name & age.

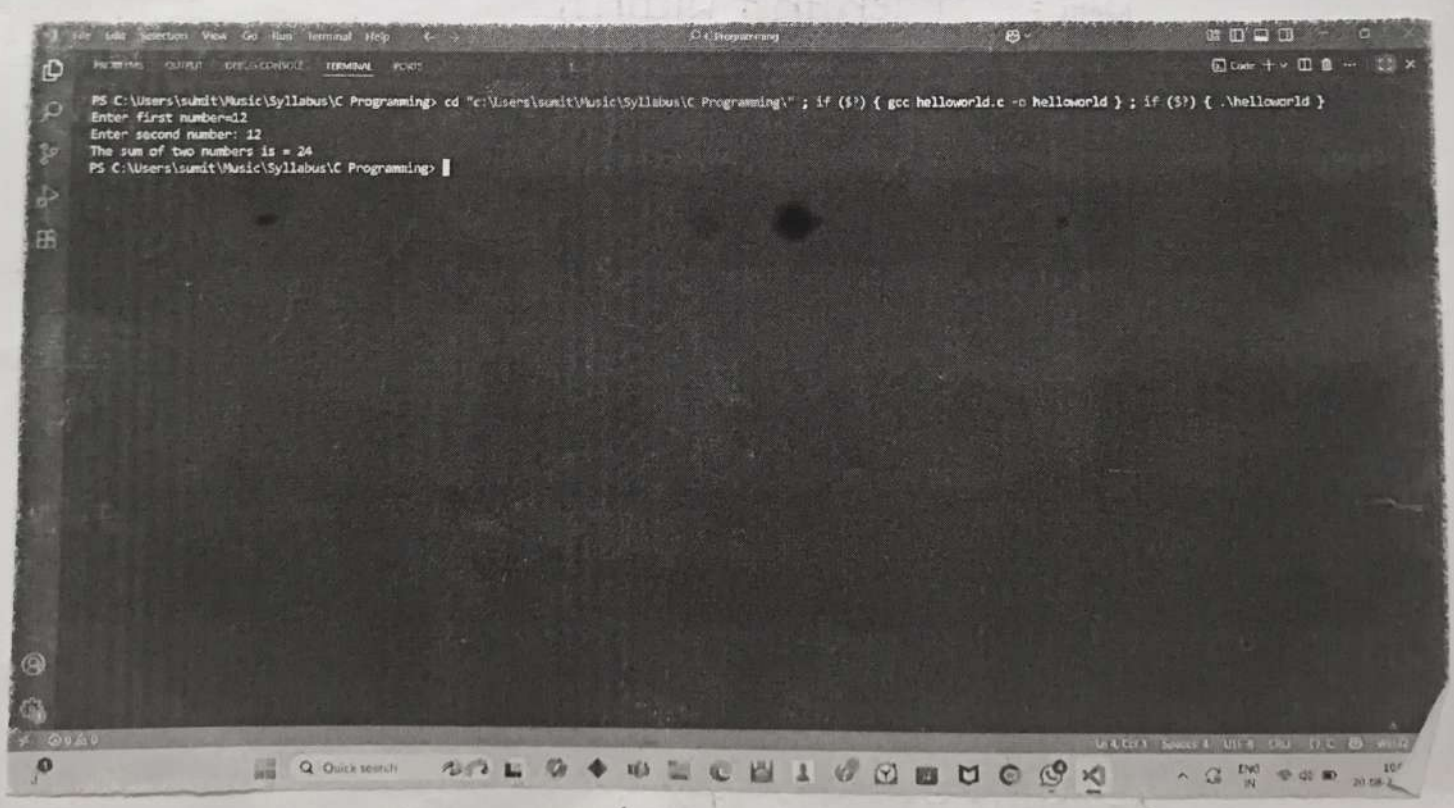
```
#include <stdio.h>
int main() {

  int age;
  printf("Enter your name: ");
  scanf("%s",&name);
  printf("Enter your age: ");
  scanf("%d",&age);
  printf("Your name is %s and You are %d years old\n",name,age);
  return 0;
}
```

4. Write a C program to add two numbers, take number from user.

```
#include <stdio.h>
int main(){
  int a, b, sum;
  printf("Enter first number: ");
  scanf("%d",&a);
  printf("Enter second number: ");
  scanf("%d",&b);
  sum = a+b;
  printf("The sum of two numbers is = %d ", sum);
  return 0;
}
```

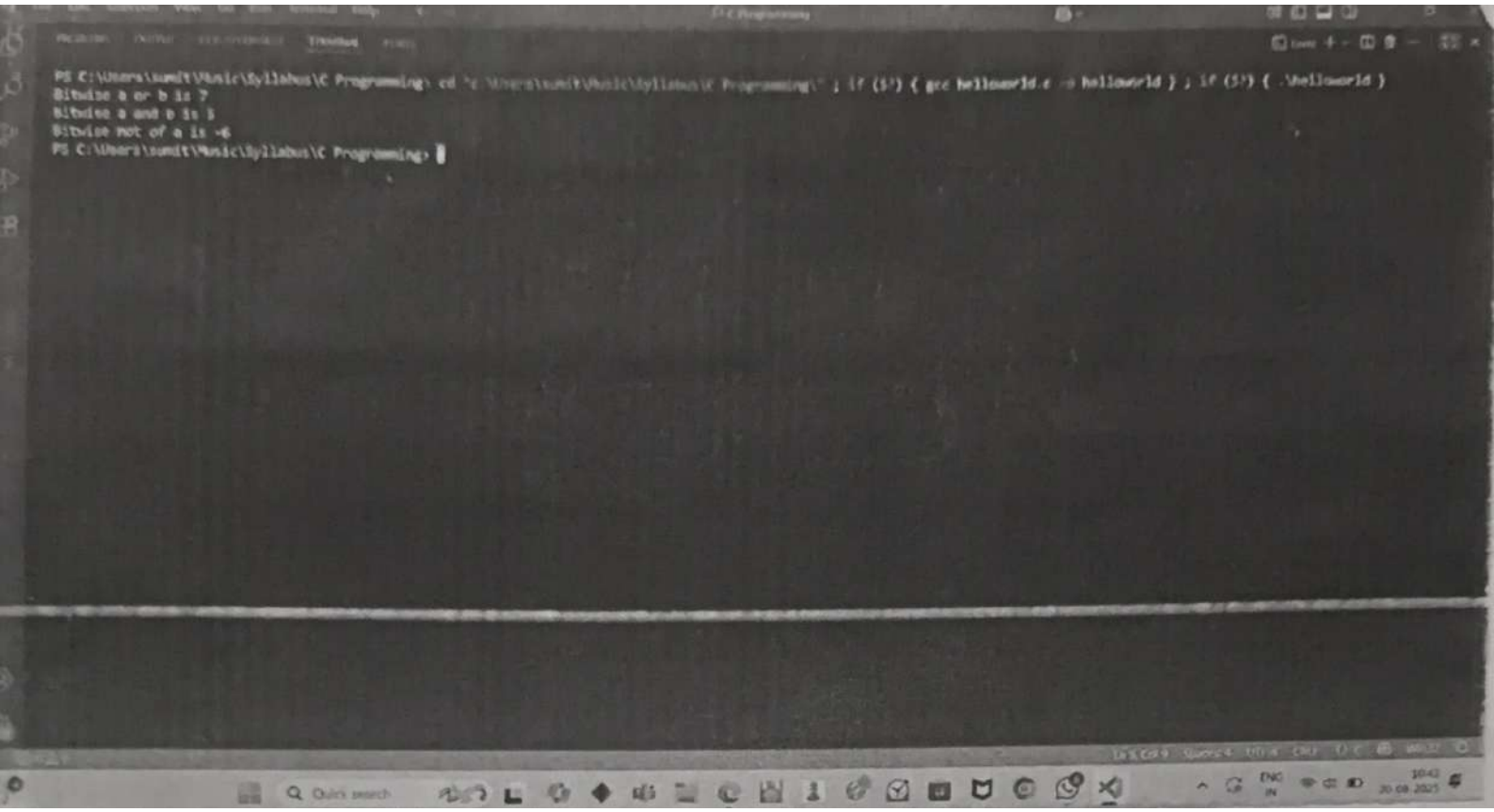

PS C:\Users\sumit\Music\Syllabus\C Programming> cd "c:\Users\sumit\Music\Syllabus\C Programming\" ; if ($?) { gcc helloworld.c -o helloworld } ; if ($?) { .\helloworld }
Enter your name: vedans
Enter your age: 19
Your name is vedans
Your age is 19
PS C:\Users\sumit\Music\Syllabus\C Programming>


PS C:\Users\sumit\Music\Syllabus\C Programming> cd "c:\Users\sumit\Music\Syllabus\C Programming\" ; if ($?) { gcc helloworld.c -o helloworld } ; if ($?) { .\helloworld }
Enter first number=12
Enter second number: 12
The sum of two numbers is = 24
PS C:\Users\sumit\Music\Syllabus\C Programming>

EXPERIMENT NO. 11: BITWISE OPERATOR

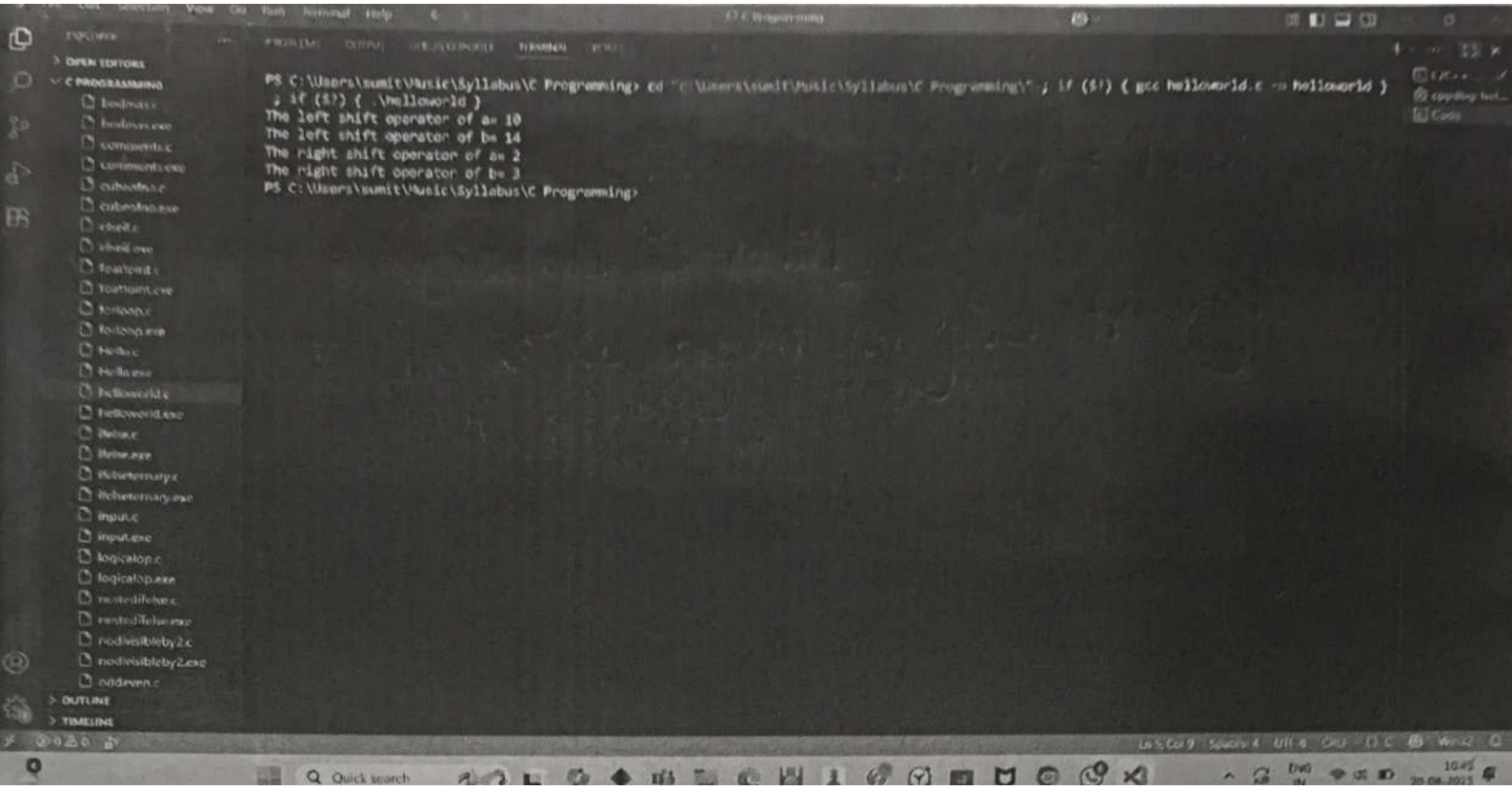
1. Write a C program to apply bitwise OR, AND, NOT operators on bit level.

```
# include <stdio.h>
  int main () {
  // bit level operator
  int a = 5;
  int b = 7;
  printf ("Bitwise a orb is %d \n", a|b);
  printf ("Bitwise a andb is %d \n", a&b);
  printf ("Bitwise not of a is %d \n", ~a);
  return 0;
}
```


PS C:\Users\sumit\Music\Syllabus\C Programming\ cd "c:\Users\sumit\Music\Syllabus\C Programming\" ; if ($?) { gcc helloworld.c -o helloworld } ; if ($?) { .\helloworld }
Bitwise a or b is 7
Bitwise a and b is 1
Bitwise not of a is -6
PS C:\Users\sumit\Music\Syllabus\C Programming>

2. Write a program to apply left shift & right shift operator.

```
#include <stdio.h>
int main() {
int a=5;
int b=7;
printf("The left shift operator of a %d \n", a<<1);
printf("The left shift operator of b %d \n", b<<1);
printf("The right shift operator of a %d \n", a>>1);
printf("The right shift operator of b %d \n", b>>1);
return 0;
}
```


PS C:\Users\sumit\Music\Syllabus\C Programming> cd "c:\Users\sumit\Music\Syllabus\C Programming\" ; if ($?) { gcc helloworld.c -o helloworld } ; if ($?) { .\helloworld }
The left shift operator of a= 10
The left shift operator of b= 14
The right shift operator of a= 2
The right shift operator of b= 3
PS C:\Users\sumit\Music\Syllabus\C Programming>

EXPERIMENT: 3.1: CONDITIONAL STATEMENTS

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

Prog:

```
#include <stdio.h>
int main(){
    float a,b,c;
    printf("Enter three sides of triangle");
    scanf("%f %f %f", &a, &b, &c);
    if((a+b>c) && (a+c>b) && (b+c>a))
    printf("Triangle is valid"\n");
    if(a==b && b==c)
    printf("It is an equilateral triangle\n");
    else if(a=>b || b=>c || a==c)
    printf("It is an isosceles triangle\n");
    else
    printf("It is a scalene triangle\n");
    if((a*a==b*b+c*c) || (b*b==a*a+c*c) ||
       (c*c==a*a+b*b))
    printf("It is also a right angled triangle\n");
    else
    printf("Triangle is not valid\n");
    return 0;
}
```

1. Output :-

? Enter three sides of triangle = 12cm, 9cm, 6cm
? Triangle is valid
? It is a right angled triangle

2. WAP to compute the BMI of a person & print BMI value as per the following ranges.

$$BMI = Weight(Kg) / Height^2(mts)$$

Prog:-

```
#include <stdio.h>
int main() {
float weight, height, BMI;
    printf("Enter the weight");
    scanf("%f", &weight);
    printf("Enter the height");
    scanf("%f", &height);
    BMI = weight / (height * height);
    printf("Your BMI is = %2f \n", BMI);
    if (BMI < 15)
    printf("Category : Starvation \n");
    else if (BMI >= 15.1 && BMI <= 17.5)
    printf("Category : Under Weight \n");
    else if (BMI >= 17.6 && BMI <= 24.9)
    printf("Category : Ideal \n");
    else if (BMI >= 25 && BMI <= 25.9)
    printf("Category : Obese \n");
    else if (BMI >= 30 && BMI <= 39.9)
    printf("Category : Morbidly Obese \n");
    return 0;
}
```

2. Output :

? Enter Weight (in kg) = 70
? Enter height (in meters) = 1.75

Ans. Your BMI is = 22.86
Category = Ideal

CLASSMATE
Date :
Page :

EXPERIMENT: 3.2: LOOPS

1. W.A.P to enter no. till the user wants. At the end, it should display the count of positive, negative & zeros entered.

```
#include <stdio.h>
int main() {
    int num;
    int positive_count = 0;
    int negative_count = 0;
    int zero_count = 0;
    char choice;
    do {
        printf("Enter a number: ");
        scanf("%d", &num);
        if (num > 0) {
            positive_count++;
        }
        else if (num < 0) {
            negative_count++;
        }
        else {
            zero_count++;
        }
        printf("Do you want to enter another no.? (y/n)");
        scanf("%c", &choice);
}
while (choice == 'y' || choice == 'Y');
printf("Count of positive numbers: %d \n", positive_count);
```

Output :-

```
Enter a number : 2
Do you want to enter another number ? (y/n) : y
Enter a number : -4
Do you want to enter another number ? (y/n) : y
Enter a number : 0
Do you want to enter another number ? (y/n) : n
Count of positive number : 1
Count of negative number : 1
Count of zeros : 1
```

```
printf("Count of negative numbers: %d\n", negative_count);
printf("Count of zeros: %d\n", zero_count);
return 0;
}
```

2. W.A.P to print the multiplication table for a user-entered number, formatted as num * i = result.

```
#include <stdio.h>
int main(){
        int num, i;
        printf("Enter a number:");
        scanf("%d", num);
        printf("Multiplication Table of %d:\n", num);
        for(i=1; i<=10; i++){
                printf("%d * %d = %d\n", num, i, num*i);
        }
        return 0;
}
```

Output:

Enter a number: 5

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

3. Write a program to generate the following set of output:

Output :-

```
   1
  2 3
 4 5 6
```

```
# include <stdio.h>
int main ( ){
        int num=1;
        for (int i=1; i<=4; i++){
            for (int k=1, k<=4-i; k++){
                    printf (" ");
            }
            for(int j=1; j<=i; j++ ){
                    printf ("%d", num);
                    num = num+1;
            }
            printf("\n");
        }
        return 0;
}
```

4. The population of a town is 100000. The population has increased steadily at a rate of 10% for 10 years. WAP to determine the population at each year in the last decade.

```
#include <stdio.h>
int main(){
    float population = 100000;
    float rate = 0.10;
    int years = 10;
    for(int i=1; i<=years; i++){
        population = population * (1+rate);
        printf("End of year %d population will be =
            %d", i, population);
        }
    return 0;
}
```

5. 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 upto a reasonable limit. Example of Ramanujan number: 1729
$12^3 + 1^3 + 10^3 + 9^3$. for a number L=20 (that is limit)

```
#include <stdio.h>
#include <math.h>
#define LIMIT 100000
int main(){
    int count_sums[LIMIT] = {0};
    int sum_pairs[LIMIT][2] = {0};
    int i, j, sum;
    for (i = 1; i < 47; i++){
        sum = i * i * i + j * j * j;
        if (sum < LIMIT){
            if (count_sums[sum] == 1){
                printf("%d = %d^3 + %d^3 and %d^3 + %d^3 \n",
                sum, sum_pairs[sum][0], sum_pairs[sum][1], i,
                );
            }
            if (count_sums[sum] == 0){
                sum_pairs[sum][0] = i;
                sum_pairs[sum][1] = j;
            }
            count_sums[sum]++;
        }
    }
}
return 0;
}
```

OUTPUT:

Enter the limit (L): 20

Ramanujan numbers upto cube of 20:

$1729 = 1^3 + 12^3 = 9^3 + 10^3$

$4104 = 2^3 + 16^3 = 9^3 + 15^3$

CLASSMATE
Date :
Page :

EXPERIMENT : 4

1. Declare a global variable outside all functions & use it inside various functions to understand its accessibility.

```
#include <stdio.h>
int global_variable = 100;
void modify_global(){
        global_variable = 200;
        printf("Inside modify_global(): global_variable = %d\n",
                global_variable);
        }
        void accessGlobal(){
                printf("Inside accessGlobal(): global_variable = %d\n",
                        global_variable);
                }
        int main(){
                printf("Inside main(): Initial global_variable = %d\n",
                        global_variable);
                modifyGlobal();
                printf("Inside main(): global_variable after modify
                        Global() call = %d\n", global_variable);
                accessGlobal();
                global_variable = 30;
                printf("Inside main(): global_variable after direct modification
                        = %d\n", global_variable);
                return 0;
        }
```

OUTPUT:

Inside main(): Initial global_variable = 100

Inside. modify_global(): global variable = 200

Inside main(): global_variable after modify Global() call • 200

Inside access Global(): global_variable = 200

Inside main(): global_variable after direct modification = 30

2. Declare a local variable inside a function & try to access it outside the function. Compare this with accessing the global variable from within the function.

```
#include <stdio.h>
int global_var = 10;
void my_function() {
        int local_var = 20;
        printf("Inside my_function:\n");
        printf("Accessing global_var: %d\n", global_var);
        printf("Accessing local_var: %d\n", local_var);
}
int main() {
    printf("Inside main:\n");
    printf("Accessing global_var: %d\n", global_var);
    my_function();
    return 0;
}
```

OUTPUT:

Inside main :

Accessing global_var : 10

Inside my_function :

Accessing global_var : 10

Accessing local_var : 20

3. Declare variable within different code blocks (enclosed by curly braces) & test their accessibility within & outside those blocks.

```
#include <stdio.h>
int main ( ) {
    int global_to_main = 10;
    printf ("Inside main block: global_to_main = %d \n", global_to_main);
    {
    int inner_block_var1 = 20;
    printf ("Inside inner block 1: global_to_main = %d, inner_block_var1
        = %d \n", global_to_main, inner_block_var1 );

    int inner_block_var2 = 30;
    printf ("Inside inner block 2; global_to_main = %d, inner_block_var1 =
        %d, iner_block_var2 = %d \n", global_to_main, inner_block_var1
        , inner_block_var2 );
    }
return 0;
}
```

Output :

Inside main block : global_to_main = 10

Inside inner block 1 : global_to_main = 10

inner_block_var1 = 20

4. Declare a static local variable inside a function. Observe & how its value across function calls.

```
#include <stdio.h>
void counterFunction(){
    static int callcount = 0;
    callcount++;
    printf("function called %d times \n", callcount);
}
int main(){
    counterFunction();
    counterFunction();
    counterFunction();
    return 0;
}
```

OUTPUT:

```
function called 1 times.
function called 2 times
function called 3 times.
```

EXPERIMENT - 5

1. W.A.P to read a list of integers & store it in a single dimensional array. Write a C program to print the second largest integer in a list of integers.

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

int main(){
    int n;
    printf("Enter the no. of elements in array:");
    scanf("%d",&n);
    if(n<2){
    printf("Invalid input: The array must contain atleast two
            elements to find the second largest.\n");
    return 1;
}
int arr[n];
printf("Enter %d integers :\n",n);
for(int i=0;i<n;i++){
    printf("Element %d:",i+1);
    scanf("%d",&arr[i]);
    }
int firstlargest = INT_MIN;
int secondlargest= INT_MIN;

for(int=0; i<n; i++){
    if(arr[i]>firstlargest){
```

```
            secondlargest = firstlargest;
            firstlargest = arr[i];
        }
        else if (arr[i] > secondlargest && arr[i] != firstlargest) {
            secondlargest = arr[i];
        }
    }
    if (secondlargest == INT_MIN) {
        printf("There is no distinct second largest element");
    }
    else {
        printf("The secondlargest element in the array is: %d\n"
            , secondlargest);
    }
return 0;
}
```

OUTPUT 1:

```
Enter the no. of elements in array : 5
Enter 5 integers :

Element 1 : 10
Element 2 : 20
Element 3 : 5
Element 4 : 40
Element 5 : 30

The second largest element in the array is: 30
```

2. W.A.P to read a list of integers & store it in a single dimensional array. W.A.P to count & display positive, negative, odd & even numbers in an array.

```
#include <stdio.h>
int main () {
    int size, i;
    int posi_count = 0;
    int nega_count = 0;
    int odd_count = 0;
    int even_count = 0;
    printf ("Enter size of array: ");
    scanf ("%d", &size);
    int arr[size];
    printf ("Enter %d integers : \n", size);
    for (i=0, i<size, i++) {
        scanf ("%d", &arr[i]);
    }
    for (i=0; i< size; i++) {
        if (arr[i] > 0) {
            posi_count ++;
        }
        else if (arr[i] < 0) {
            nega_count ++;
        }
        if (arr[i] %2 == 0) {
            even_count ++;
        }
        else {
            odd_count ++;
        }
```

2. W.A.P to read a list of integers & store it in a single dimensional array. W.A.P to count & display positive, negative, odd & even numbers in an array.

```
#include <stdio.h>
int main () {
    int size, i;
    int posi_count = 0;
    int nega_count = 0;
    int odd_count = 0;
    int even_count = 0;
    printf ("Enter size of array: ");
    scanf ("%d", &size);
    int arr[size];
    printf ("Enter %d integers : \n", size);
    for (i=0, i<size, i++) {
        scanf ("%d", &arr[i]);
    }
    for (i=0; i< size; i++) {
        if (arr[i] > 0) {
            posi_count ++;
        }
        else if (arr[i] < 0) {
            nega_count ++;
        }
        if (arr[i] % 2 == 0) {
            even_count ++;
        }
        else {
            odd_count ++;
        }
```

OUTPUT:

```
Enter the size of array : 6
Enter 6 integers :
5 -3 0 8 -2 7

Positive numbers : 3
Negative numbers : 2
Odd numbers : 3
Even numbers : 3
```

```
}
printf("Positive numbers : %d \n", posi_count);
printf("Negative numbers : %d \n", nega_count);
printf("Even numbers : %d \n", even_count);
printf("Odd numbers : %d \n", odd_count);
return 0;
}
```

3. WAP to find frequency of a particular no. in a list of integers.

```
#include <stdio.h>
int main(){
    int n, i, search, count = 0;
    int arr[100];
    printf("Enter no. of Elements : ");
    scanf("%d", &n);
    printf("Enter %d numbers : \n", n);
    for(i=0; i<n; i++){
        scanf("%d", &arr[i]);
    }
    printf("Enter numbers to find frequency : ");
    scanf("%d", &search);
    for(i=0; i<n, i++){
        if(arr[i] == search)
        count++;
    }
    printf("Frequency of %d = %d times \n", search, count);
    return 0;
}
```

OUTPUT:

```
Enter the number of element in array:
Enter 6 integers:
    Enter Element 1 : 4
    Enter Element 2 : 2
    Enter Element 3 : 4
    Enter Element 4 : 7
    Enter Element 5 : 1
    Enter Element 6 : 4
Enter the no. to find its frequency : 4
The number 4 appears 3 times in the array.
```

4. WAP that reads two matrices A(m×n) & B(p×q) & computes the product A & B. Read matrix A & B in row major order respectively. Print both the input matrices & resultant matrix. With suitable headings & output should be in matrix format only. Program must check the compatibility of orders of the matrices for multiplication. Report appropriate message in case of incompatibility.

```
#include <stdio.h>
int main(){
    int m,n,p,q,i,j,k;
    printf("Enter the order of matrix A(m n):\n");
    scanf("%d %d",&m,&n);
    int a[m][n];
    printf("Enter elements in matrix A:\n");
    for(i=0; i<m; i++){
        for(j=0; j<n; j++){
            scanf("%d",&a[i][j]);
        }
    }
printf("Enter the order of matrix B(p q):\n");
scanf("%d %d",&p,&q);

int b[p][q];
printf("Enter the elements in matrix B:\n");
for(i=0; i<p; i++){
    for(j=0; j<q; j++){
        scanf("%d",&b[i][j]);
    }
}
```

```
if(n!=p){
    printf("Matrix multiplication is not possible.\n");
    printf("Reason: No. of columns of A(%d) != No. of rows of B
            (%d)".    );
    return 0;
}

int result[m][q];
for(i=0; i<m; i++){
    for(j=0; j<q; j++)
    {
        result[m][q]=0;
        for(k=0; k<n; k++){
            result[i][j]=result[i][j]+a[i][k]*b[k][j];
        }
    }
}
printf("\n Matrix A:\n");
for(i=0; i<m; i++){
    for(j=0; j<n; j++){
        printf("%d", a[i][j]);
    }
}
printf("\n Matrix B:\n");
for(i=0; i<p; i++){
    for(j=0; j<q; j++){
        printf("%d", b[i][j]);
    }
}
printf("\n Resultant matrix:\n");
```

```
for(i=0; i<m; i++){
    for(j=0; j<q; j++){
        printf("%d", result[i][j]);
    }
    printf("\n");
}
return 0;
}
```

OUTPUT:

```
Enter rows and columns of matrix A (m n): 2 2
Enter rows and columns of matrix B (p q): 2 2
Enter elements of matrix A (2x2) in row-major order:
1
2
3
4
Enter elements of Matrix B (2x2) in row-major order:
5
6
7
8
Matrix A:
1   2
3   4
Matrix B:
5   6
7   8
Product Matrix (A x B):
19   22
43   50
```

CLASSMATE
Date :
Page :

EXPERIMENT : 6

1. Develop a recursive & non-recursive function FACT(num) to find the factorial of a number, n! defined by FACT(n) = 1, if n=0. Otherwise, FACT(n) = n * FACT(n-1). Using this function, Write a C program to compute the binomial coefficient. Tabulate the results for different values of n & r with suitable messages.

```
#include <stdio.h>
int FACT (int);
int bincoeff (int n, int r);
int main (){
    int n,r, num;
    printf ("Enter a number :\n");
    scanf ("%d", &num);
    int a = FACT (num);
    printf ("Factorial of a number : %d \n", a);
    printf ("Enter the value of n & r : \n");
    scanf ("%d %d", &n, &r);
    float z = bincoeff (n,r);
    printf ("Binary coefficiant of a number : %.1f", z)
    return 0;
}
int FACT (int num)
if (num == 0){
    return 1;
}
else {
    return num * FACT (num-1);
```

Date

```
int bincoeff(int n, int r){
    float c = (FACT(n)) / (FACT(r) * (FACT(n-r)));
    return c;
}
```

2. Develop a recursive function GCD(num1, num2) that accept two integer arguments. WAP that invokes this function to find the greatest common divisor of two given integers.

```
#include <stdio.h>
int GCD(int num1, int num2);
int main(){
    int num1, num2;
    printf("Enter two numbers :\n");
    scanf("%d %d", &num1, &num2);
    int a = GCD(num1, num2);
    printf("Greatest common divisior of two given
            integers : %d \n", a);
    return 0;
}
int GCD(int num1, int num2){
    if (num2 == 0)
    return num1;
    else
        return GCD(num2, num1 % num2);
}
```

Mohan

OUTPUT:

```
Enter a number:
5
Factorial of a number: 120
Enter the value of n & r:
4 2
Binary coefficient of a number: 6.0
```

OUTPUT:

```
Enter two numbers:
12 18
Greatest common divisor for two given integers: 6
```

Date

3. 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.

```
#include <stdio.h>
int FIBO (int num);
int main (){
    int i, a, num;
    printf ("Enter the limit upto which fibo series
            print : ");
    scanf ("%d", &num);
    printf ("Fibonacci series : ");
    for (i=0 ; i<num ; i++ ){
        int a = FIBO (i);
        printf = ("%d", a );
    }
    int FIBO (int num){
        if (num==0)
            return 0;
        if (num==1).
            return 1;
        else
            return FIBO (num-2) + FIBO(num-1);
    }
}
```

Mohan

OUTPUT:

Enter the limit upto which Fibo series print: 6

Fibonacci Series: 0 1 1 2 3 5

Date

4. Develop a function REVERSE (str) that accepts a string argument. WAP that invokes this function to find the reverse of a given string.

```
#include <stdio.h>
#include <string.h>
void REVERSE (str[]){
    int i, len;
    char temp;
    len = strlen (str);
    for (i=0; i< len/2; i++) {
        temp = str[i];
        str[i] = str[len-i-1];
        str[len-i-1] = temp;
    }
}
int main (){
    char str[100];
    printf ("Enter a string :");
    scanf ("%[^\n]", str);

    REVERSE (str);
    printf ("Reversed string :%s \n", str);
    return 0;
}
```

OUTPUT:

Enter a string : Hello World

Reversed string : dlroW olleH

Date

EXPERIMENT - 8

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

```
#include <stdio.h>
int main() {
    int a = 10;
    float b = 3.14;
    char c = 'x';
    int *p = &a;
    float *p1 = &b;
    char *p2 = &c;
    printf("Variable values:\n");
    printf("a = %d\n", a);
    printf("b = %.2f\n", b);
    printf("c = %c\n", c);
    printf("Pointer values:\n");
    printf("p = %p\n", p);
    printf("p1 = %p\n", p1);
    printf("p2 = %p\n", p2);
    printf("Values using dereferencing:\n");
    printf("*p = %d\n", *p);
    printf("*p1 = %.2f\n", *p1);
    printf("*p2 = %c\n", *p2);
    return 0;
}
```

OUTPUT:

Variable values:

a = 10

b = 3.14

c = x

Pointer values:

p1 = 0x7fff e487e564

p2 = 0x7fff e487e560

p3 = 0x 7fff e487e56f

Values using dereferencing:

*p1 = 10

*p2 = 3.14

*p3 = x

Date

2. Perform pointer arithmetic (increment & decrement) on pointers of different data types. Observe how the memory address change & the effects on data access.

```
#include <stdio.h>
int main (){
  int a[3] = {10, 20, 30};
  float b[3] = {1.1, 2.2, 3.3};
  char c[3] = {'A', 'B', 'C'};
  int *p1 = a;
  float *p2 = b;
  char *p3 = c;
  printf ("Original addresses :\n");
  printf ("p1 = %p\n", p1);
  printf ("p2 = %p\n", p2);
  printf ("p3 = %p\n", p3);
  p1++; p2++; p3++;
  printf ("After Increment :\n");
  printf ("p1 = %p, %d\n; p1, *p1);
  printf ("p2 = %p, %.2f\n", p2, *p2);
  printf ("p3 = %p, %c\n", p3, *p3);
  p1--;
  p2--;
  p3--;
  printf ("\n After Decrement\n");
  printf ("p1 = %p, %d\n", p1, *p1);
  printf ("p2 = %p, %.1f\n", p2, *p2);
  printf ("p3 = %p, %c\n", p3, *p3);
  return 0;
}
```

Mohan

OUTPUT:

Original Addresses:

p1 = 0x7fff90e81bec

p2 = 0x7fff90e81be0

p3 = 0x7fff90e81bdd

After Increement:

p1 = 0x7fff90e81bf0, 20

p2 = 0x7fff90e81be4, 2.2

p3 = 0x7fff90e81bde, B

After Decreement (back to original):

p1 = 0x7fff90e81bec, 10

p2 = 0x7fff90e81be0, 1.1

p3 = 0x7fff90e81bdd, A

Date

3. Write a function that accepts pointers as parameters. Pass variables by reference using pointers & modify their values within the function.

→ When we pass variable as pointers, the function can directly modify their values in memory.

```
#include <stdio.h>
void swap(int *x, int*y){
    int temp = *x;
        *x = *y;
        *y = temp;
}
int main()
{
int a=5, b=10;
printf("Before swap\n");
printf("a= %d, b= %d\n",a,b);

swap(&a, &b);
printf("\n After swap :\n");
printf("a= %d, b= %d\n",a,b);
return 0;
}
```

OUTPUT:

Before swap:

a = 5, b = 10

After swap:

a = 10, b = 5

Date

EXPERIMENT - 7

1. W.A.P that uses functions to program the following operations:
   a. Reading a complex number
   b. Writing a complex number
   c. Addition & Subtraction of two complex numbers.

```
#include <stdio.h>
struct complex
{
    float real;
    float imag;
};

struct complex readcomplex()
{
    struct complex c;
    printf("Enter real part: \n");
    scanf("%f", &c.real);
    printf("Enter imaginary part: ");
    scanf("%f", &c.imag);
    return c;
}
void writecomplex(struct complex c)
if (c.imag >=0)
    printf("%.2f + %.2f i \n", c.real, c.imag);
else
```

Date

```
    printf("%.2f - %.2f i\n", c.real, c.imag);
}
struct complex addcomplex (struct complex a, struct
                           complex b)
{
    struct complex result;
    result.real = a.real + b.real;
    result.imag = a.imag + b.imag;
    return result;
}
struct complex subtractcomplex (struct complex a,
                                struct complex b)
{
    struct complex result;
    result.real = a.real - b.real;
    result.imag = a.imag - b.imag;
    return result;
}
int main(){
    struct complex c1, c2, sum, diff;
    printf("Enter first complex number:\n");
    c1 = readcomplex();
    printf("Enter second complex number:\n");
    c2 = readcomplex();
    sum = addcomplex(c1, c2);
    diff = subtractcomplex(c1, c2);
    printf = ("\n First Complex Number:");
    writecomplex(c1);
    printf("\n Second Complex Number:");
    writecomplex(c2);
```

Mohan

Date

```
printf ("In Addition:");
writecomplex (sum);
printf ("Subtraction:");
writecomplex (diff);
return 0;
}
```

Date

2. W.A.P to compute the monthly pay of 100 employees using each employer name, basic pay. The DA is computed as 52% of the basic pay. Gross salary (basic pay + DA). Print the employers name & gross salary.

```
#include <stdio.h>
struct employee {
    char name[50];
    float basic_pay;
    float gross_salary;
};
int main() {
    struct employee emp[3];
    int i;
    printf("Enter details of 3 employees:\n");
    for (i=0; i<3; i++) {
        printf("\n Employee %d Name:", i+1);
        scanf("%s", emp[i].name);
        printf("Basic pay:");
        scanf("%f", &emp[i].basic_pay);
        float da = 0.52 * emp[i].basic_pay;
        emp[i].gross_salary = emp[i].basic_pay + da;
        printf("\n---- Employee Gross Salary list ----\n");
        for (i=0; i<3; i++) {
            printf("Name: %s \t Gross Salary: %.2f\n",
                emp[i].name, emp[i].gross_salary);
    };
    return 0;
}
```

Mohan

Date

3. Create a book structure containing book_id, title, author name & price. Write a C programme to pass a structure as a function argument & print the book details.

```
#include <stdio.h>
struct Book {
    int book_id;
    char title[50];
    char author[50];
    float price;
};

void displayBook(struct Book b) {
    printf("\n Book Details :\n");
    printf("ID : %d \n", b.book_id);
    printf("Title : %s \n", b.title);
    printf("Author : %s \n", b.author);
    printf("Price : %.2f \n", b.price);
}
int main() {
    struct Book b1;
    printf("Enter Book ID:");
    scanf("%d", &b1.book_id);
    printf("Enter Title:");
    scanf("%s", &b1.title);
    printf("Enter Author :");
    scanf("%s", &b1.author);
    printf("Enter Price :");
    scanf("%f", &b1.price);
    displayBook(b1);
    return 0;
}
```

Date

4. Create a union containing six strings: name, home address, hostel_address, city, state & zip. Write a C program to display your present address.

```
#include <stdio.h>
union Address {
    char name[50];
    char home_address[100];
    char hostel_address[100];
    char city[50];
    char state[50];
    char zip[50];
};
int main() {
    union Address addr;
    printf("Enter Name: ");
    scanf("%s", addr.name);
    printf("Name : %s \n", addr.name);

    printf("Enter Home Address : ");
    scanf("%s", addr.home_address);
    printf("Enter Hostel Address : ");
    scanf("%s", addr.hostel_address);
    printf("Hostel Address : %s \n", addr.hostel_address);

    printf("Enter City : ");
    scanf("%s", addr.city);
    printf("City : %s \n", addr.city);
```

Date

4. Create a union containing six strings: name, home_address, hostel_address, city, state & zip. Write a C program to display your present address.

```
#include <stdio.h>
union Address {
    char name[50];
    char home_address[100];
    char hostel_address[100];
    char city[50];
    char state[50];
    char zip[50];
};
int main() {
    union Address addr;
    printf("Enter Name: ");
    scanf("%s", addr.name);
    printf("Name : %s \n", addr.name);

    printf("Enter Home Address : ");
    scanf("%s", addr.home_address);
    printf("Enter Hostel Address : ");
    scanf("%s", addr.hostel_address);
    printf("Hostel Address : %s \n", addr.hostel_address);

    printf("Enter City : ");
    scanf("%s", addr.city);
    printf("City : %s \n", addr.city);
```

Date

```
printf ("Enter state : ");
scanf ("%s", addr.state);
printf ("state : %s \n", addr.state);

printf ("Enter ZIP code : ");
scanf ("%s", addr.zip);
printf ("ZIP code : %s \n", addr.zip);
return 0;
}
```

Date

EXPERIMENT - 9

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

```
#include <stdio.h>
int main () {
    File * fp;
    char text [200];
    fp = fopen ("myfile.txt", "w");
    if (fp == NULL) {
        printf ("Error creating file !");
        return 1;
    }
    printf ("Enter text to write into the file : \n");
    fgets ( text, sizeof (text), stdin);
    fputs (text, fp );
    fclose (fp);
    printf ("File created and written successfully ! \n");
    return 0;
}
```

Date

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

```
#include <stdio.h>
int main() {
    FILE *fp;
    char ch;
    fp = fopen("myfile.txt", "r");
    if (fp == NULL) {
        printf("File not found!");
        return 1;
    }
    printf("File content :\n");
    return 1;
}
printf("File content :\n");
while((ch = fgetc(fp)) != EOF)
{
fput char(ch);
}
fclose(fp);
return 0;
}
```

Date

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

```
#include <stdio.h>
int main() {
    FILE * fp;
    char line[200]
    fp = fopen ("myfile.txt", "r");
    if (fp == NULL) {
        printf ("File Not found!");
        return 1;
    }
    printf ("Reading file line by line : \n");
    while (fgets (line, size of (line), fp));
    {
    printf ("%s", line);
    }
    fclose (fp);
    return 0;
}
```

Date

EXPERIMENT = 10

1. Write a program to create a simple linked list in C using pointer and structure.

```
#include <stdio.h>
#include <stlib.h>
 struct Node
{
 int data;
 Struct Node * next;
};
 int main()
 {
     Struct Node * head = NULL,
   * second = NULL, * third = NULL;

head =(Struct Node*)malloc(sizeof(Struct Node));
second = (struct Node*)malloc(size of(struct Node));
third = (struct Node*)malloc(size of(Struct Node));
  head → data =10;
  head → next = second;
  second → data = 20;
  second → next = third;
  third → data = 30;
  third → next = NULL;
 printf("Linked List: ");
 struct Node *temp =head;
```

Date

```
while (temp != NULL)
{
printf("%d → ", temp → data);
temp = temp → next;
}
print ("NULL \n");
return 0;
}
```

Date

2. Write a program to insert item in middle of the linked list.

```
# include <stdio.h>
# include <stdlib.h>
 Struct Node
 {
   int data;
   Struct Node *next;
 };

 Void insertMiddle (struct Node * head,
            int data, int position)
 {
  struct Node * newNode
      = (struct Node*) malloc (size of
                  (struct Node));
  newNode → data = data;
  Struct Node * temp = head;
  int i;
  for (i = 1; i < position; i++)
   {
    if (temp == NULL)
   {
    printf ("Position out of range!
   return;
  }
```

Date

```
    temp = temp -> next;
}

newNode -> next = temp -> next;
temp -> next = newNode;
}
int main()
{
struct Node *head = NULL, *second = NULL;
 * third = NULL;
head = (struct Node*) malloc(sizeof(struct Node));
second = (struct Node *) malloc(size of(structNode));
third = (struct Node *) malloc(size of(struct Node));

head -> data = 10;
head -> next = second;
second -> data = 20;
second -> next = third;
third -> data = 30;
third -> next = NULL;
printf("Original list: 10 -> 20 -> 30 -> NULL\n");
insertMiddle(head, 15, 1);
printf("Updated list:");
struct Node * temp = head;
while (temp != NULL) {
        printf("%d -> ", temp -> data);
        temp = temp -> next;
}
printf("NULL\n");
}
```

Date

EXPERIMENT - 12

1. W.A.P to define some constant variable in preprocessor.

```
#include <stdio.h>
#define PI 3.14
#define MAX 100
#define MIN 1

int main() {
    printf("PI = %.2f \n", PI);
    printf("MAX = %d \n", MAX);
    printf("MIN = %d \n", MIN);
    return 0;
}
```

2. WAP to define a function in directives.

```
#include <stdio.h>
#define SQUARE(x) ((x)*(x))
int main() {
    int num = 5;
    printf("Square of %d is %d \n", num, SQUARE(num));
    return 0;
}
```

Date

EXPERIMENT - 13

1. W.A.P to define multiple macro to perform arithmetic functions.

```
#include <stdio.h>
#define ADD (a,b)   ((a) + (b))
#define SUB (a,b)   ((a)-(b))
#define MUL (a,b)   ((a) * (b))
#define DIV (a,b)   ((a) / (b))
int main () {
    int x = 20 , y = 4;
    printf ("Addition = %d \n", ADD (x,y));
    printf ("Subtraction = %d \n", SUB (x,y));
    printf ("Multiplication = %d \n", MUL (x,y));
    printf ("Division = %d \n", DIV (x,y));
    return 0;
}
```

Mohan

Date

2. Open an existing file & read its content character by character & then close the file.

```
#include <stdio.h>
int main(){
    FILE *fp;
    char ch;
    fp = fopen("myfile.txt", "r");
    if (fp == NULL){
        printf("File not found !");
        return 1;
    }
    printf("File content :\n");
    while ((ch = fgetc(fp)) != EOF){
        putchar(ch);
    }
    fclose(fp);
    return 0;
}
```

Date

EXPERIMENT - 14

Q1. W.A.P to create a static library for programming arithmetic functions.

arith.h

```
int add (int, int);
int sub (int, int);
int mul (int, int);
int div (int, int);
```

arith.c

```
int add (int a, int b)
{
    return a + b;
}
int sub ( int a, int b)
{
    return a-b;
}
int mul (int a, int b)
{
   return a * b;
}
int div (int a, int b)
{
    return a / b;
}
```

Mohan

Date

Steps to create static library :

1 Compile the source file

gcc -c arith.c

2 Create the static library

ar rcs libarith.a arith.o

3. Static library created

libarith.a

Date

2. W.A.P to use static library in other program.

FILE : main_static.c

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

int main(){
    int a = 10, b = 5
    printf("Add = %d \n", add(a,b));
    printf("Sub = %d \n", sub(a,b));
    printf("Mul = %d \n", mul(a,b));
    printf("Div = %d \n", divide(a,b));
    return 0;
}
```

Compilation

```
gcc main_static.c -L. -larith -o static_app
```

RUN

```
./static_app
```

Date

EXPERIMENT - 15

1. W.A.P to create a shared library for programing arithmatic functions.

File 1 : arith_shared.h

```
int add (int, int);
int sub (int, int);
int mul (int, int);
int div (int, int);
```

File 2 : arith_shared.C

```
int add (int a, int b)
{
return a+b;
}
int sub (int a, int b)
{

return a-b;
} int mul (int a, int b)
return a*b;
}
int divide (int a, int b)
{
return a/b;
}
```

Date

Steps to create shared library

1. Compile with position-independent code

   gcc -fPIC -c arith_shared.c

2. Create shared library

   gcc -shared -o libarith.so arith_shared.o

3. Shared library created

   libarith.so

Date

2. W.A.P to use shared library in other program.

FILE : main_shared.c

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

int main () {
    int a=10, b=5;
    printf ("Add = %d \n", add (a,b));
    printf ("Sub = %d \n", sub (a,b));
    printf ("Mul = %d \n", mul (a,b));
    printf ("Div = %d \n", divide (a,b));

    return 0;
}
```

Compilation

```
gcc main_shared.c -L. -larith -O shared_app
```

Run

```
export LD_LIBRARY_PATH=.
./shared_app
```