SRI RAMAKRISHNA MISSION VIDYALAYA COLLEGE OF ARTS AND SCIENCE

(An Autonomous Institution Affiliated to Bharathiar University,
Re-Accredited by NAAC with A+grade)

COIMBATORE-641 020

DEPARTMENT OF COMPUTER APPLICATIONS

RECORD NOTE

Core Practical: Object Oriented Programming in C++
Subject Code: (23PCA1CP1)

This is certified that this is a bonafide record of work done by

Name:____________________________ **Reg .No.:**____________________________

Staff-In-Charge **Head of the Department**

Submitted for the Practical Examination held at Sri Ramakrishna Mission Vidyalaya College of Arts and Science on __________________ during the academic year 2024-2025.

Examiners

Internal **External**

INDEX

EX.No: 1	**1.Write a C++ program for Prime Numbers.**
DATE:	

Aim:

To find the given number reverse than the given number is prime or not.

Algorithm:

Step 1: To start the program.

Step 2: Using the input statement and print the number.

Step 3: Check the conditional statement and values is not equal zero.

Step 4: The using operator and calculate the reverse number.

Step 5: Print the reverse number.

Step 6: Next reverse number prime or not prime check the conditional statement.

Step 7: Print the statement prime or not.

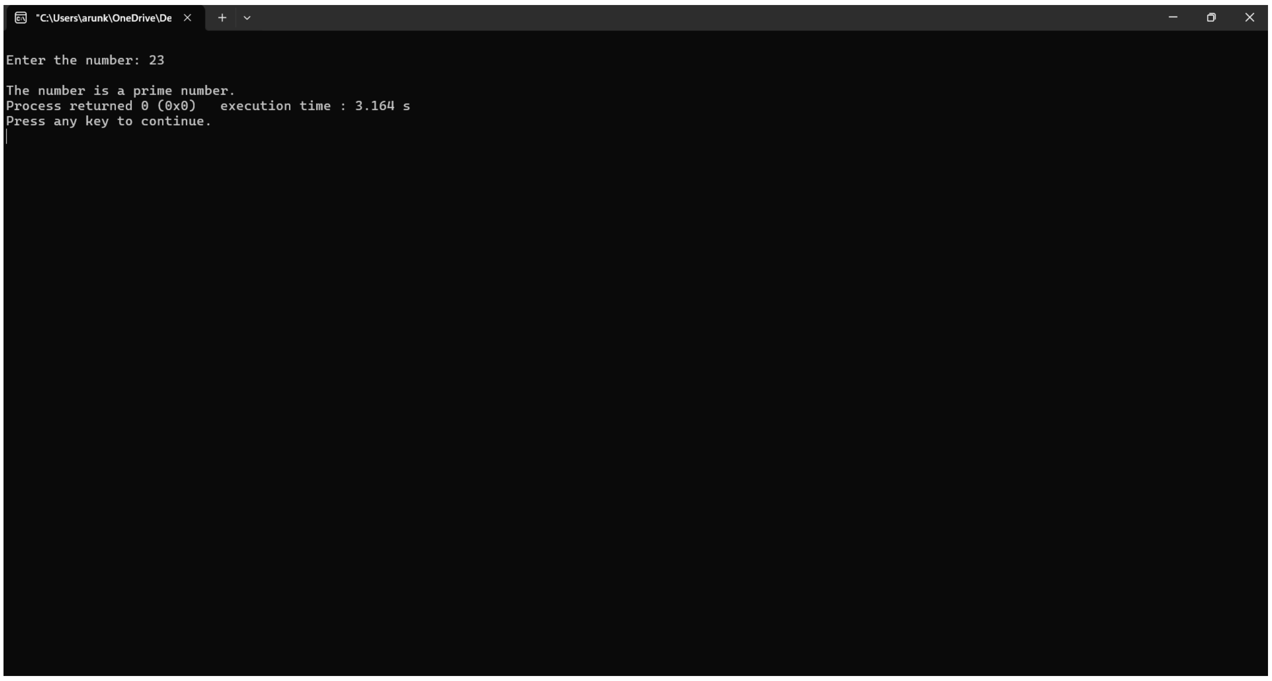
Step 8: End the program.

1. Write a C++ program for Prime Numbers.

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cout << "\nEnter the number: ";
    cin >> n;

    bool isPrime = true;
    if (n <= 1) {
        isPrime = false;
    } else {
        for (int i = 2; i <= n / 2; ++i) {
            if (n % i == 0) {
                isPrime = false;
                break;
            }
        }
    }

    if (isPrime) {
        cout << "\nThe number is a prime number.";
    } else {
        cout << "\nThe number is not a prime number.";
    }

    return 0;
}
```

EX.No: 2	**2.Write a C++ program to find the maximum and minimum numbers from a list of integers.**
DATE:	

<u>Output:</u>



```
Enter the number: 23

The number is a prime number.
Process returned 0 (0x0)   execution time : 3.164 s
Press any key to continue.
```

<u>RESULT:</u>

Thus the Above Program was Executed Successfully.

Aim:

To find the program to check maximum & minimum numbers.

Algorithm:

Step 1 : To start the program

Step 2 : Print the range of set of number

Step 3 : To check the conditional statement

Step 4 : Print the maximum and minimum numbers

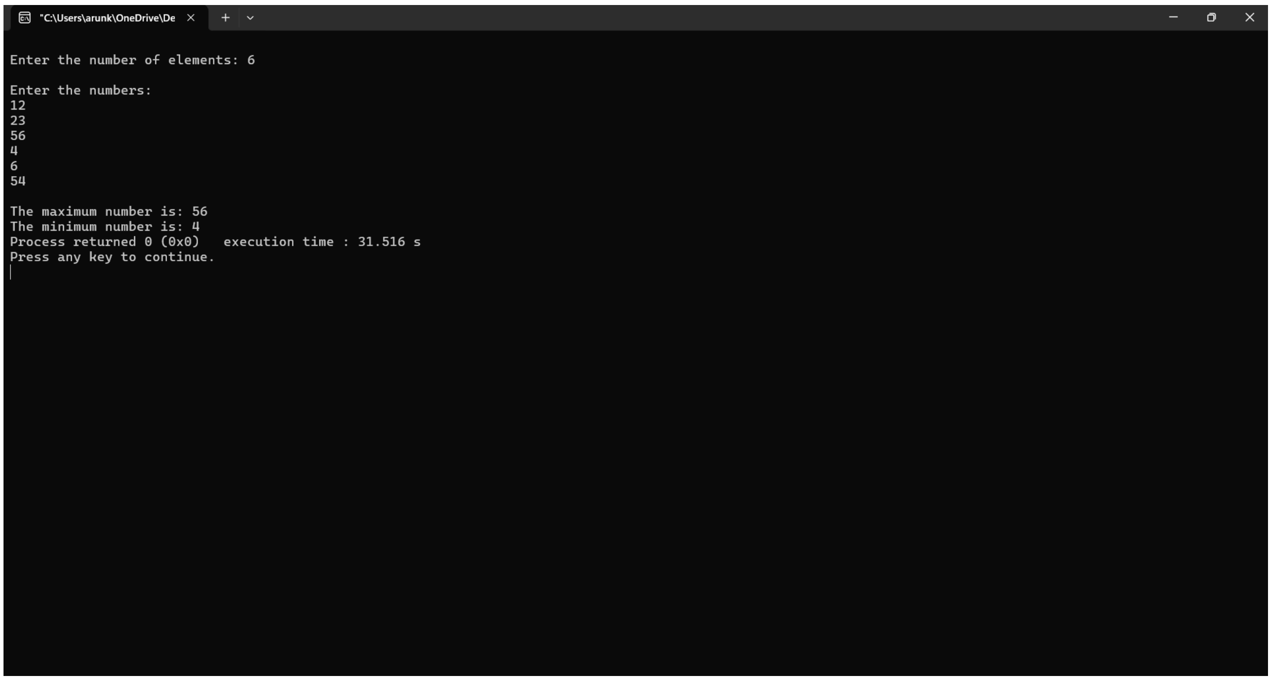
Step 5 : End the program

2.Write a C++ program to find the maximum and minimum numbers from a list of integers.

```
#include <iostream>
using namespace std;

int main() {
    int i, j, n, t, a[100];

    cout << "\nEnter the number of elements: ";
    cin >> n;

    cout << "\nEnter the numbers:\n";
    for (i = 0; i < n; i++) {
        cin >> a[i];
    }

    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++) {
            if (a[i] < a[j]) {
                t = a[i];
                a[i] = a[j];
                a[j] = t;
            }
        }
    }

    cout << "\nThe maximum number is: " << a[0];
    cout << "\nThe minimum number is: " << a[n - 1];

    return 0;
}
```

EX.No: 3	**Write a C++ program to calculate the retirement date based on the user's current age and the retirement age.**
DATE:	

Output:



```
Enter the number of elements: 6

Enter the numbers:
12
23
56
4
6
54

The maximum number is: 56
The minimum number is: 4
Process returned 0 (0x0)   execution time : 31.516 s
Press any key to continue.
```

RESULT:

Thus the Above Program was Executed Successfully.

Aim:

To find the Retirement Date for the employee.

Algorithm:

Step 1: To start the program

Step 2: Enter the details of the employee while joining

Step 3: Check the today's date

Step 4: Using the conditional statement the age and date of Retirement employee

Step 5: Display the details of the employee date of retirement

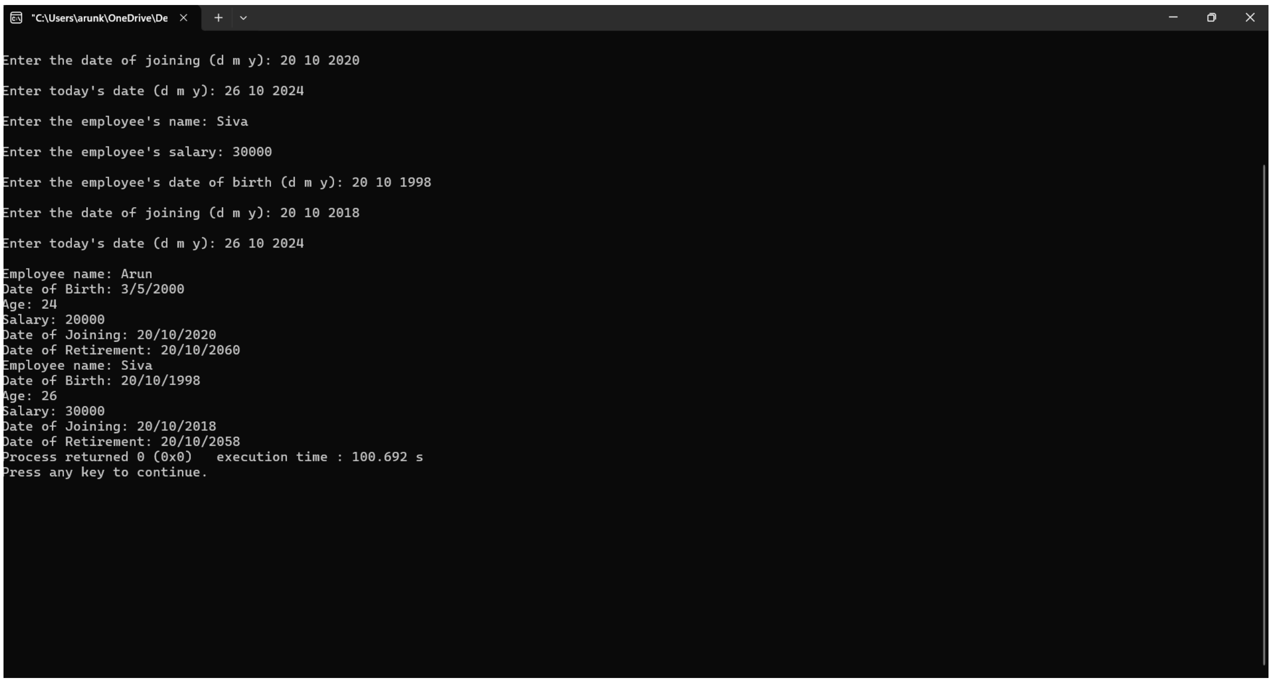
Step 6: End the program

3. Write a C++ program to calculate the retirement date based on the user's current age and the retirement age.

```
#include <iostream>
using namespace std;

class Emp {
    char name[20];
    int age, salary;
    int d, m, y; // Date of birth
    int d1, m1, y1; // Date of joining
    int d2, m2, y2; // Today's date

public:
    void input() {
        cout << "\nEnter the employee's name: ";
        cin >> name;
        cout << "\nEnter the employee's salary: ";
        cin >> salary;
        cout << "\nEnter the employee's date of birth (d m y): ";
        cin >> d >> m >> y;
        cout << "\nEnter the date of joining (d m y): ";
        cin >> d1 >> m1 >> y1;
        cout << "\nEnter today's date (d m y): ";
        cin >> d2 >> m2 >> y2;

        // Calculate age
        if (m > m2 || (m == m2 && d > d2)) {
            age = y2 - y - 1;
        } else {
            age = y2 - y;
        }
    }

    void display() {
        cout << "\nEmployee name: " << name;
        cout << "\nDate of Birth: " << d << "/" << m << "/" << y;
        cout << "\nAge: " << age;
        cout << "\nSalary: " << salary;
        cout << "\nDate of Joining: " << d1 << "/" << m1 << "/" << y1;
        cout << "\nDate of Retirement: " << d1 << "/" << m1 << "/" << (y + 60);
    }
};

int main() {
    Emp e[10];
    int n;

    cout << "Enter the total number of employees: ";
    cin >> n;

    for (int i = 0; i < n; i++) {
        e[i].input();
    }

    for (int i = 0; i < n; i++) {
        e[i].display();
    }

    return 0;
}
```

EX.No: 4	**Write a C++ program to sort a list of strings in alphabetical order.**
DATE:	

Output:



"C:\Users\arunk\OneDrive\De

```
Enter the date of joining (d m y): 20 10 2020

Enter today's date (d m y): 26 10 2024

Enter the employee's name: Siva

Enter the employee's salary: 30000

Enter the employee's date of birth (d m y): 20 10 1998

Enter the date of joining (d m y): 20 10 2018

Enter today's date (d m y): 26 10 2024

Employee name: Arun
Date of Birth: 3/5/2000
Age: 24
Salary: 20000
Date of Joining: 20/10/2020
Date of Retirement: 20/10/2060
Employee name: Siva
Date of Birth: 20/10/1998
Age: 26
Salary: 30000
Date of Joining: 20/10/2018
Date of Retirement: 20/10/2058
Process returned 0 (0x0)   execution time : 100.692 s
Press any key to continue.
```

RESULT:

Thus the Above Program was Executed Successfully.

Aim:

To write a program to sort a set of string and display it.

Algorithm:

```
Step 1 : Start the program
Step 2 : Enter the set of string
Step 3 : Accept the string
Step 4 : Using the conditional statement and handling function
Step 5 : Display the string in sorted form
Step 6 : End the program
```

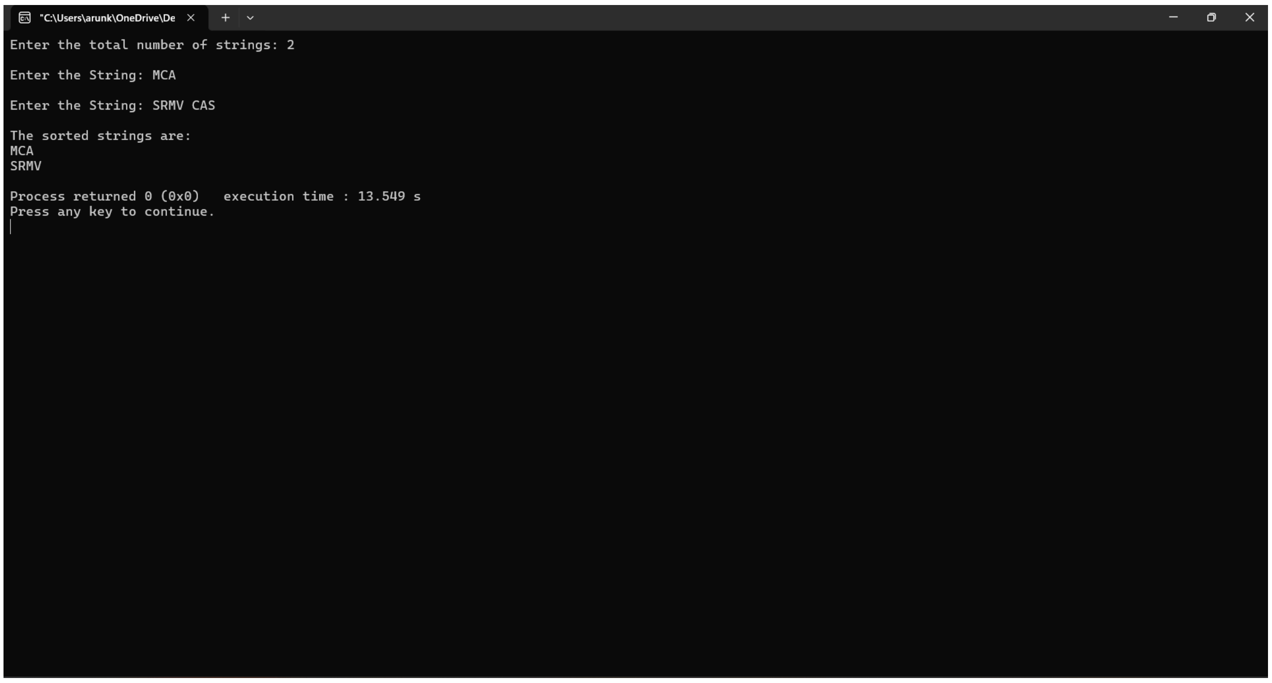
4. Write a C++ program to sort a list of strings in alphabetical order.

```
#include <iostream>

#include <cstring>

using namespace std;


class Sort {

public:

   char s[100];  //A character array (C-string) to store a string of up to 100
characters.


   // Function to input a string

   void getstring() {

      cout << "\nEnter the String: ";

      cin >> s;

   }555

};


int main() {

   int n;

   char temp[100];
```

```
// Ask user for the number of strings
cout << "Enter the total number of strings: ";
cin >> n;

Sort a[100];  // Array of objects

// Get strings from the user
for (int i = 0; i < n; i++) {
   a[i].getstring();
}

// Sort the strings using bubble sort
for (int i = 0; i < n; i++) {
   for (int j = i + 1; j < n; j++) {
      if (strcmp(a[i].s, a[j].s) > 0) {  //  Compare two strings.
         strcpy(temp, a[i].s);         // Copy one string to another.
         strcpy(a[i].s, a[j].s);
         strcpy(a[j].s, temp);
      }
   }
}
```

```
    // Display the sorted strings
    cout << "\nThe sorted strings are: \n";
    for (int i = 0; i < n; i++) {
        cout << a[i].s << endl;
    }

    return 0;
}
```

Output:



"C:\Users\arunk\OneDrive\De

```
Enter the total number of strings: 2

Enter the String: MCA

Enter the String: SRMV CAS

The sorted strings are:
MCA
SRMV

Process returned 0 (0x0)   execution time : 13.549 s
Press any key to continue.
```

RESULT:

Thus the Above Program was Executed Successfully.

EX.No: 5	**Write a C++ program to implement binary search on a sorted array of integers.**
DATE:	

Aim:

To write a program to sort the set of number and find the position sorted form.

Algorithm:

Step 1: Start the program

Step 2: Enter the number of elements and enter the one by one

Step 3: Check the condition

Step 4: Calculate the position of the elements

Step 5: Display the output

Step 6: End the program

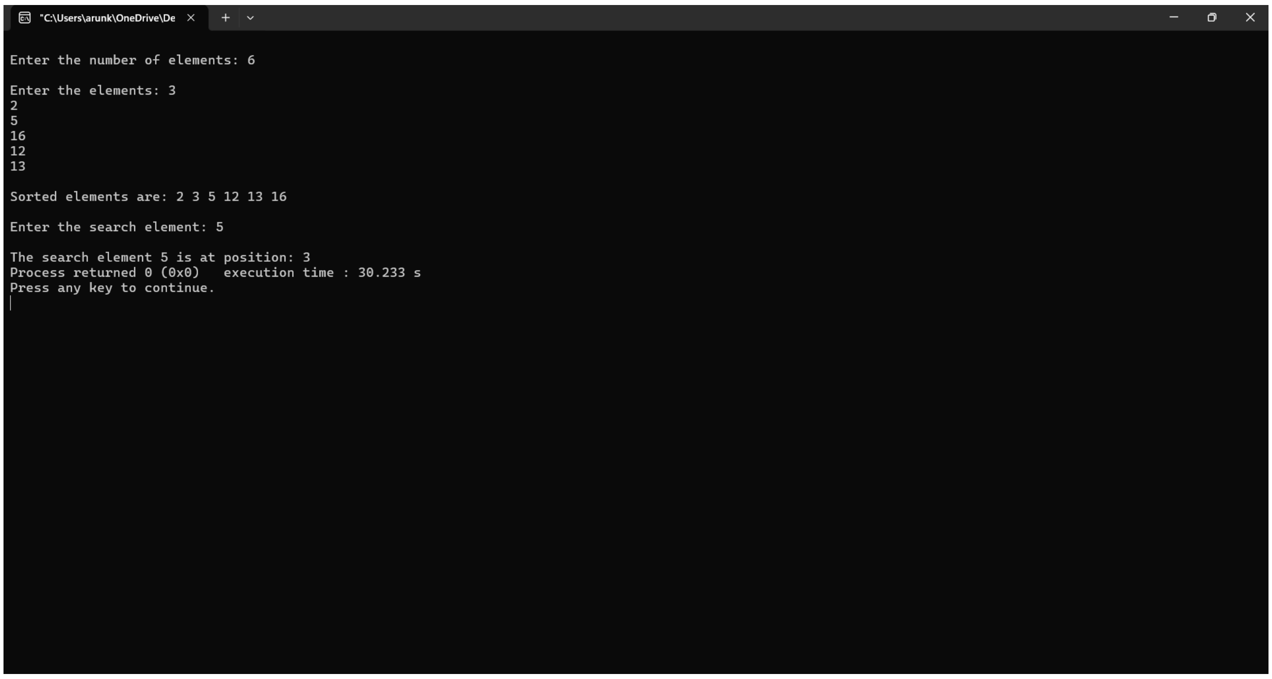
5. Write a C++ program to implement binary search on a sorted array of integers.

```
#include<iostream>

using namespace std;

class BinarySearch {

private:

    int a[20], i, t, j, s, n;

public:

    // Function to input data

    void data() {

        cout << "\nEnter the number of elements: ";

        cin >> n;

        cout << "\nEnter the elements: ";

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

            cin >> a[i];

        }

    }

    // Function to sort the array in ascending order

    void asc() {

        cout << "\nSorted elements are: ";

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

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

                if (a[i] > a[j]) {

                    t = a[i];

                    a[i] = a[j];
                    a[j] = t;
                }
            }
        }
    }
    // Function to display the sorted array
    void display() {
        for (i = 0; i < n; i++) {
            cout << a[i] << " ";
        }
        cout << endl;
    }

    // Function to perform binary search
    void search() {
        cout << "\nEnter the search element: ";
        cin >> s;

        int m, l = 0, u = n - 1, got = 0;

        while (l <= u) {
            m = (l + u) / 2;
            if (s == a[m]) {
                cout << "\nThe search element " << s << " is at position: " << m + 1;
                got = 1;
```

```
                break;
            }
            else if (s < a[m]) {
                u = m - 1;
            }
            else {
                l = m + 1;
            }
        }

        if (got != 1) {
            cout << "\nThe search element was not found.";
        }
    }
};

int main() {
    BinarySearch p1;
    p1.data();      // Get the data
    p1.asc();       // Sort the array
    p1.display();   // Display the sorted array
    p1.search();    // Perform binary search

    return 0;
}
```

EX.No: 6	**Write a C++ program to handle time in hours and minutes.**
DATE:	

Output:



```
"C:\Users\arunk\OneDrive\De

Enter the number of elements: 6

Enter the elements: 3
2
5
16
12
13

Sorted elements are: 2 3 5 12 13 16

Enter the search element: 5

The search element 5 is at position: 3
Process returned 0 (0x0)   execution time : 30.233 s
Press any key to continue.
```

RESULT:

Thus the Above Program was Executed Successfully.

Aim:

To finding the adding the times.

Algorithm:

Step 1: To start the program

Step 2: Enter the time 1

Step 3: To adding two times

Step 4: Print the times after adding

Step 5: End the program

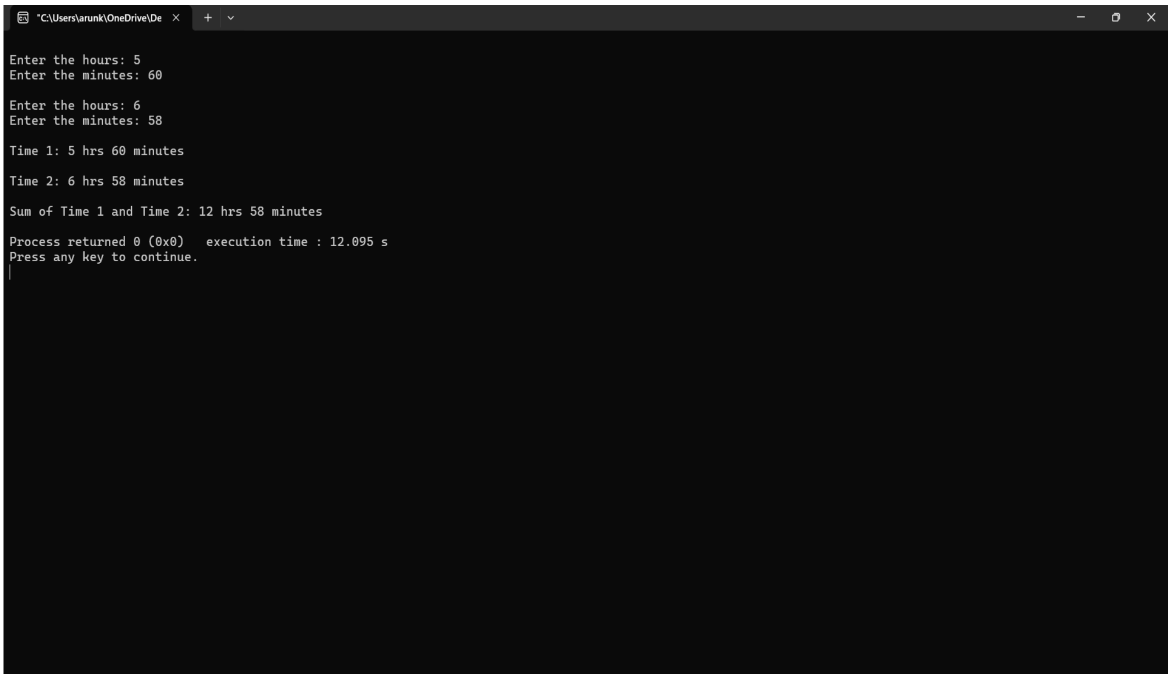
6. Write a C++ program to handle time in hours and minutes.

```
#include <iostream>
using namespace std;

class Time {
   public:
   int hrs, min;



public:
   // Function to input time
   void gettime() {
      cout << "\nEnter the hours: ";
      cin >> hrs;
      cout << "Enter the minutes: ";
      cin >> min;
   }

   // Function to display time
   void puttime() {
      cout << hrs << " hrs " << min << " minutes" << endl;
   }

   // Function to add two times
      void sum(Time t1, Time t2)
      {
      min = t1.min + t2.min;
      hrs = min / 60;  // Convert minutes to hours if needed
```

```
        min = min % 60;  // Get remaining minutes
        hrs += t1.hrs + t2.hrs;
        // Add hours
    }
};
int main() {
    Time t1, t2, t3;
    // Input two times
    t1.gettime();
    t2.gettime();
    // Sum the two times and store the result in t3
    t3.sum(t1,t2);
    // Display the times
    cout << "\nTime 1: ";
    t1.puttime();
    cout << "\nTime 2: ";
    t2.puttime();
    if(t1.min>60 || t2.min>60)
    {
        cout<<"\nInvalid input";
    }
    else
    {
    cout << "\nSum of Time 1 and Time 2: ";
    t3.puttime();
    }
    return 0;
}
```

EX.No: 7	Write a C++ program demonstrating the use of a copy constructor.
DATE:	

Output:



```
Enter the hours: 5
Enter the minutes: 60

Enter the hours: 6
Enter the minutes: 58

Time 1: 5 hrs 60 minutes

Time 2: 6 hrs 58 minutes

Sum of Time 1 and Time 2: 12 hrs 58 minutes

Process returned 0 (0x0)   execution time : 12.095 s
Press any key to continue.
```

RESULT:

Thus the Above Program was Executed Successfully.

Aim:

To find the values by using copy constructor program.

Algorithm:

Step 1 : To start the program

Step 2 : Enter the values A,B,C,D

Step 3 : Copying the values as well as by using copy constructors

Step 4 : Display the values after copying

Step 5 : End the program

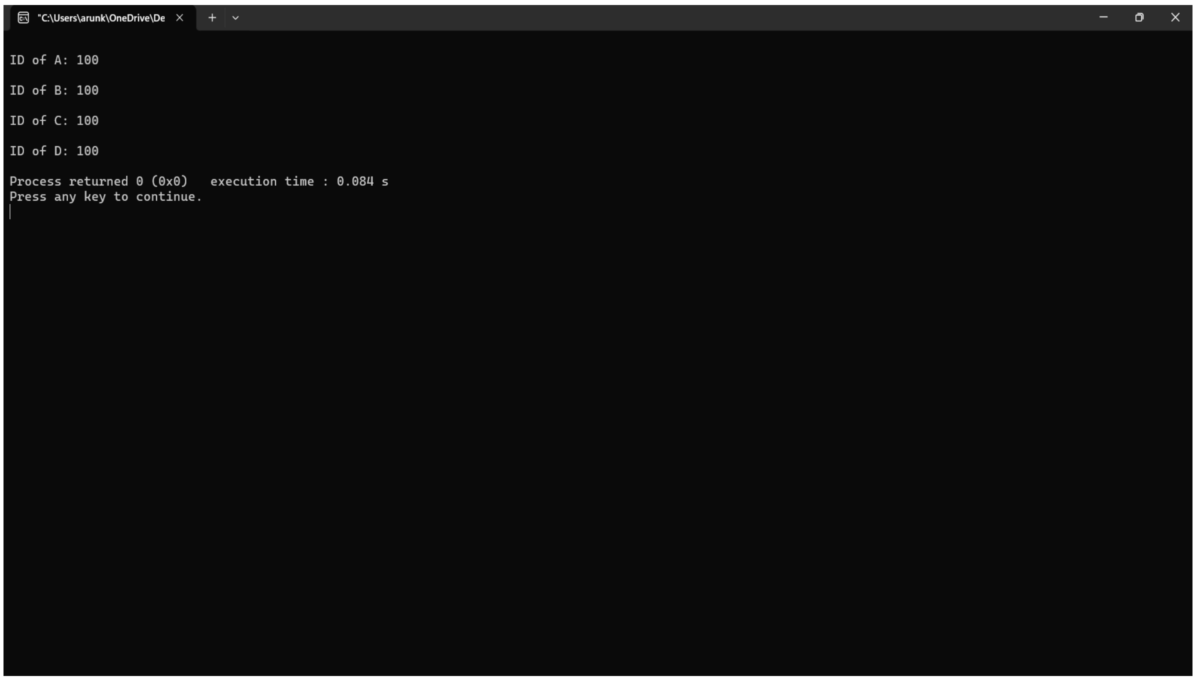
7. Write a C++ program demonstrating the use of a copy constructor.

```
#include <iostream>
```

```
using namespace std;
class code {
    int id;
public:
    // Default constructor
    code() {
        id = 0;  // Initialize id with 0
    }
    // Parameterized constructor
    code(int a) {
        id = a;
    }
    // Copy constructor
    code(const code &x) {
        id = x.id;
    }
    // Display function
    void display() {
        cout << id << endl;
    }
};
int main() {
    // Creating objects
    code A(100);  // Parameterized constructor
    code B(A);    // Copy constructor
    code C = A;   // Copy constructor (same as above)
    code D;       // Default constructor
    D = A;        // Assignment operator
    // Display the IDs
    cout << "\nID of A: ";
    A.display();
    cout << "\nID of B: ";
    B.display();
    cout << "\nID of C: ";
    C.display();
    cout << "\nID of D: ";
    D.display();
    return 0;
}
```

Output:

EX.No: 8	**Write a C++ program to calculate the factorial of a given number n.**
DATE:	



```
ID of A: 100

ID of B: 100

ID of C: 100

ID of D: 100

Process returned 0 (0x0)   execution time : 0.084 s
Press any key to continue.
```

<u>RESULT:</u>

Thus the Above Program was Executed Successfully.

Aim:

To find the factorial value of N number.

Algorithm:

Step 1 : To start the program

Step 2 : Enter the no.of values in the program

Step 3 : Declaration the factorial constructor

Step 4 : By using destructor and constructor

Step 5 : Print the answer

Step 6 : End the program

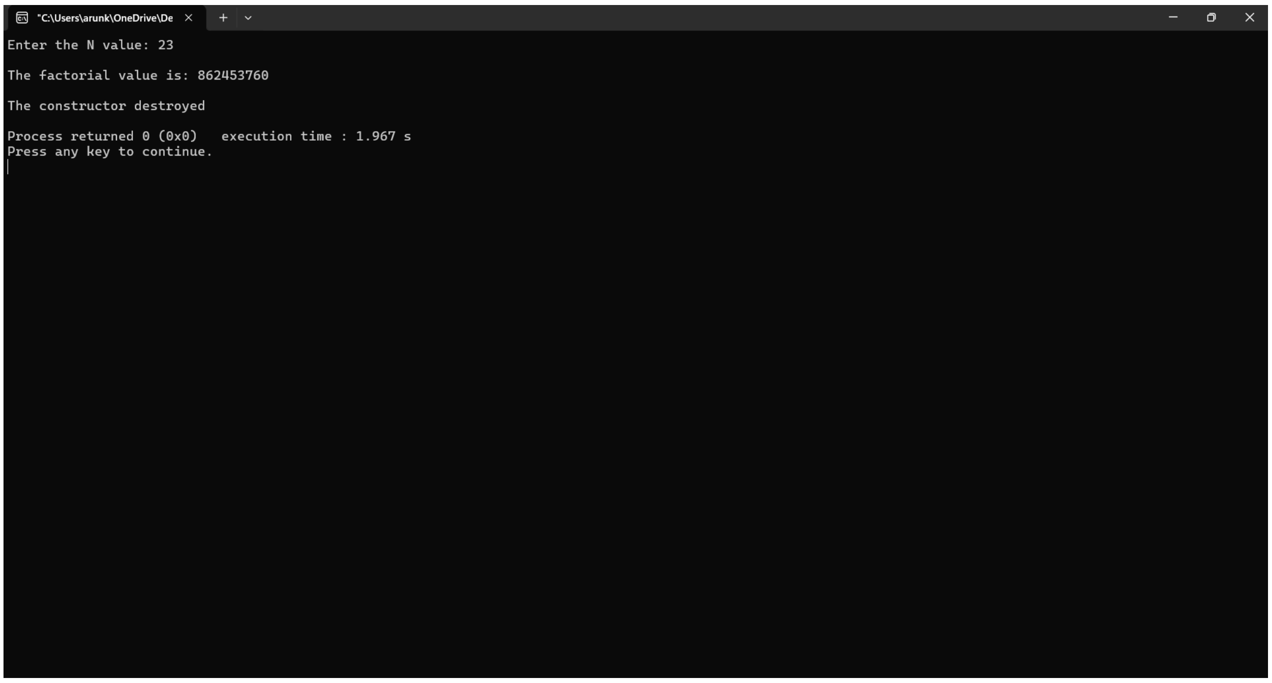
8. Write a C++ program to calculate the factorial of a given number n.

```
#include <iostream>
using namespace std;

class Factorial {
public:
   long f;  // Declare a public long variable `f` to store the factorial result.
   // Constructor to initialize the factorial with a given value.
   Factorial(int k) {
      f = k;  // Initialize `f` with the value passed (in this case, 1).
   }
   // Destructor, which will be called automatically when the object is
destroyed.
   ~Factorial() {
      cout << "\nThe constructor destroyed\n";  // This message is printed
when the object is destructed.
   }
   // Function to calculate factorial of a number `n`.
   void calculateFactorial(int n) {
      for (int i = 1; i <= n; i++) {
         f *= i;  // Multiply `f` by `i` in each iteration to calculate the factorial.
      }
      cout << "\nThe factorial value is: " << f << endl;  // Output the
calculated factorial value.
   }
};

int main() {
   int n;     // Declare an integer `n` to store the user's input number.
   long f = 1;  // Initialize the factorial variable `f` to 1 (as factorials start from
1).

   // Create a `Factorial` object with the initial value of `f`.
   Factorial f1(f);

   // Input the number for which the factorial needs to be calculated.
   cout << "Enter the N value: ";
```

```
    cin >> n;

    // Call the `calculateFactorial` function of the `Factorial` object `f1` to
compute the factorial.
    f1.calculateFactorial(n);

    return 0;
}
```

EX.No: 9	**Write a C++ Program to multiply the two values using single inheritance.**
DATE:	

Output:



"C:\Users\arunk\OneDrive\De

```
Enter the N value: 23

The factorial value is: 862453760

The constructor destroyed

Process returned 0 (0x0)   execution time : 1.967 s
Press any key to continue.
```

RESULT:

Thus the Above Program was Executed Successfully.

Aim:

To write a C++ program that demonstrates the concept of single inheritance by creating a base class for multiplication. The program will multiply two values using a derived class.

Algorithm:

```
Step 1: Start the process.
Step 2: Define a class Student that has data members to store the
        student's name and ID.
Step 3: Create a constructor in the class:
Step 4: The constructor will take parameters to initialize the
        student details (name and ID).
Step 5: Declare a member function to display the student details.
Step 6: In the main() function:
            ● Create objects of the Student class and pass values (name
              and ID) during object creation.
            ● Use the object to call the member function to display the
              student details.
Step 7: Stop the process.
```

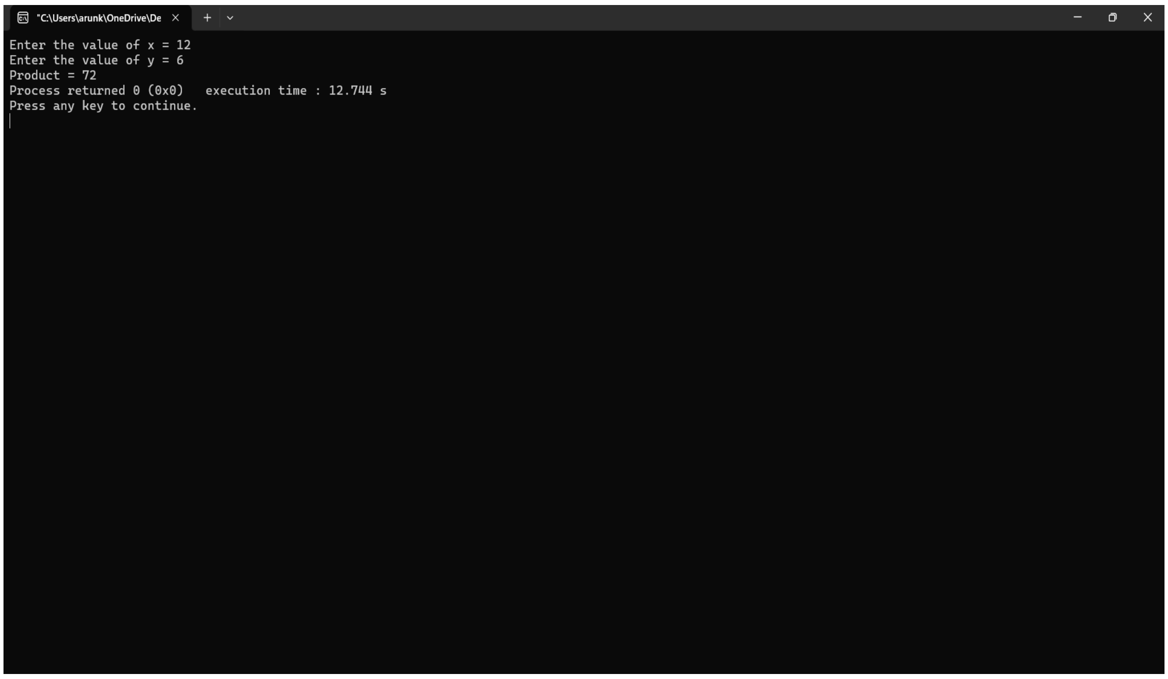
9. Write a C++ Program to multiply the two values using single inheritance.

```
#include <iostream>
```

```
using namespace std;
class base
{
  public:
    int x;
  void getdata()
  {
    cout << "Enter the value of x = ";
    cin >> x;
  }
 };
class derive : public base
{
  private:
   int y;
  public:
  void readdata()
  {
    cout << "Enter the value of y = ";
    cin >> y;
  }
  void product()
  {
    cout << "Product = " << x * y;
  }
 };
 int main()
 {
   derive a;
   a.getdata();
   a.readdata();
   a.product();
   return 0;
 }
```

Output:

EX.No: 10	**Write a C++ Program to multiply the values using multilevel inheritance.**
DATE:	



```
Enter the value of x = 12
Enter the value of y = 6
Product = 72
Process returned 0 (0x0)   execution time : 12.744 s
Press any key to continue.
```

<u>**RESULT:**</u>

Thus the Above Program was Executed Successfully.

Aim:

To write a C++ program that demonstrates the concept of multilevel inheritance by creating a hierarchy of classes to multiply two values.

Algorithm:

Step 1: Start the process.

Step 2: Define a base class Multiplier that has a member function to return the product of two values.

Step 3: Define a derived class Advanced Multiplier that inherits from Multiplier and adds a function to input two values.

Step 4: Define a further derived class Calculator that inherits from Advanced Multiplier and calls the multiplication function.

Step 5: In the main() function:

- Create an object of the Calculator class.
- Call the member function to input the values and display the result of the multiplication.

Step 6: Stop the process.

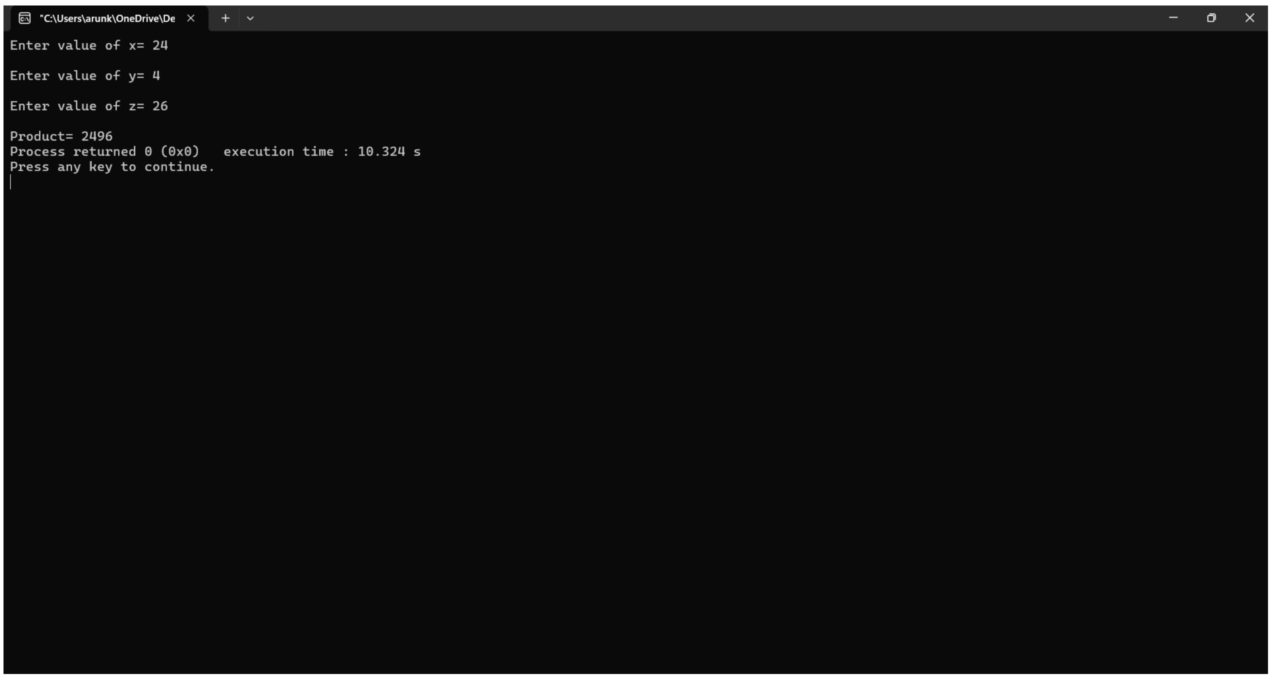
10. Write a C++ Program to multiply the values using multilevel inheritance.

```
#include <iostream>
using namespace std;
```

```
class base
{
        public:
        int x;
        void getdata()
        {
        cout << "Enter value of x= "; cin >> x;
        }
};
class derive1 : public base
{
        public:
        int y;
        void readdata()
        {
           cout << "\nEnter value of y= "; cin >> y;
        }
};
class derive2 : public derive1
{
        private:
        int z;
        public:
        void indata()
        {
        cout << "\nEnter value of z= "; cin >> z;
        }
        void product()
        {
           cout << "\nProduct= " << x * y * z;
        }
};
int main()
{
    derive2 a;
    a.getdata();
    a.readdata();
    a.indata();
    a.product();
    return 0;
}
```

Output:

EX.No: 11	**Write a C++ Program to multiply the values using hierarchical inheritance.**
DATE:	



```
Enter value of x= 24

Enter value of y= 4

Enter value of z= 26

Product= 2496
Process returned 0 (0x0)   execution time : 10.324 s
Press any key to continue.
```

RESULT:

Thus the Above Program was Executed Successfully.

Aim:

To write a C++ program that demonstrates the overloading of unary operators (++ and --). The program will show how to customize the behavior of these operators for a user-defined class.

Algorithm:

Step 1: Start the process.

Step 2: Define a base class Multiplier that has a member function to perform multiplication.

Step 3: Define two derived classes SimpleMultiplier and AdvancedMultiplier, both inheriting from Multiplier.

- SimpleMultiplier will multiply two integers.
- AdvancedMultiplier will multiply two floating-point numbers.

Step 4: In the main() function:

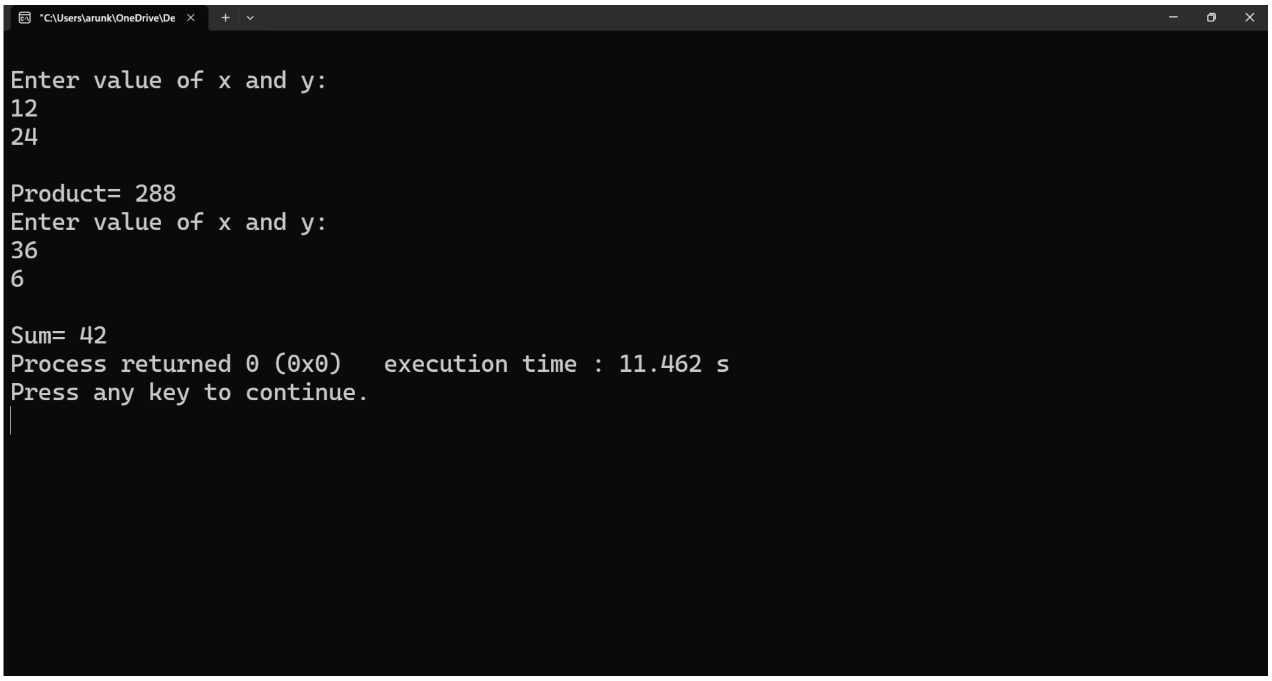
- Create objects of both derived classes.
- Call their respective multiplication functions and display the results.

Step 5: Stop the Process.

11. Write a C++ Program to multiply the values using hierarchical inheritance.

```
#include <iostream>
using namespace std;
class A
{
   public:
        int x, y;
        void getdata()
        {
           cout << "\nEnter value of x and y:\n"; cin >> x >> y;
        }
};
class B : public A
{
   public:
        void product()
        {
           cout << "\nProduct= " << x * y;
        }
};
class C : public A
{
   public:
        void sum()
        {
      cout << "\nSum= " << x + y;
        }
};
int main()
{
   B obj1;
   C obj2;
   obj1.getdata();
   obj1.product();
   obj2.getdata();
   obj2.sum();
   return 0;
}
```
Output:

EX.No: 12	Write a C++ Program to multiply the values using hybrid inheritance.
DATE:	



```
Enter value of x and y:
12
24

Product= 288
Enter value of x and y:
36
6

Sum= 42
Process returned 0 (0x0)   execution time : 11.462 s
Press any key to continue.
```

RESULT:

Thus the Above Program was Executed Successfully.

Aim:

To write a C++ program that demonstrates the concept of hybrid inheritance by combining multiple inheritance patterns.

Algorithm:

Step 1: Start the process.

Step 2: Define a base class Multiplier that has a member function to multiply two integers.

Step 3: Define two derived classes:

• SimpleMultiplier that inherits from Multiplier and implements a method to multiply two integers.

• AdvancedMultiplier that inherits from Multiplier and implements a method to multiply two floating-point numbers.

Step 4: Define a final derived class Calculator that inherits from both Simple Multiplier and AdvancedMultiplier and has methods to call the respective multiplication methods.

Step 5: In the main() function:

Step 6: Stop the process

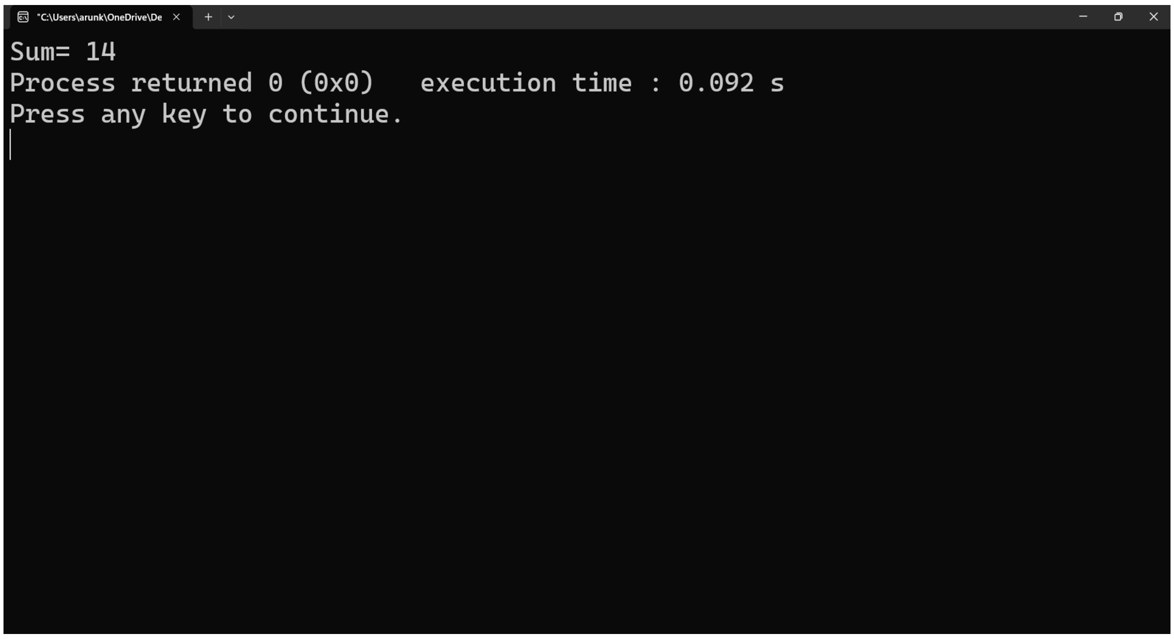
12. Write a C++ Program to multiply the values using hybrid inheritance.

```
#include <iostream>
using namespace std;
class A
{
```

```
          public:
          int x;
};
class B : public A
{
          public:
          B()
          {
            x = 10;
          }
};
class C
 {
          public:
          int y;
          C()
          {
             y = 4;
        }
};
class D : public B, public C
{
          public:
          void sum()
          {
             cout << "Sum= " << x + y;
          }
};

int main()
{
    D obj1;
          obj1.sum();
          return 0;
}
```

Output:

EX.No: 13	**Write a C++ Program to multiply the values using multiple inheritance.**
DATE:	



```
Sum= 14
Process returned 0 (0x0)   execution time : 0.092 s
Press any key to continue.
```

<u>RESULT:</u>

Thus the Above Program was Executed Successfully.

<u>Aim:</u>

To write a C++ program that demonstrates the concept of multiple inheritance by creating two base classes, each performing different operations, and a derived class that inherits from both and performs the final multiplication of values.

Algorithm:

Step 1: Start the process.

Step 2: Define the first base class Input Values that contains member functions to input two values.

Step 3: Define the second base class Multiplier that contains a member function to multiply two values.

Step 4: Define a derived class Calculator that inherits from both InputValues and Multiplier.

- In the derived class, implement a function to perform the multiplication using the values input from.
- The InputValues class and the multiplication function from the Multiplier class.

Step 5: In the main() function:

- Create an object of the Calculator class.
- Call the member functions to input values, multiply them, and display the result.

Step 6: Stop the Process.

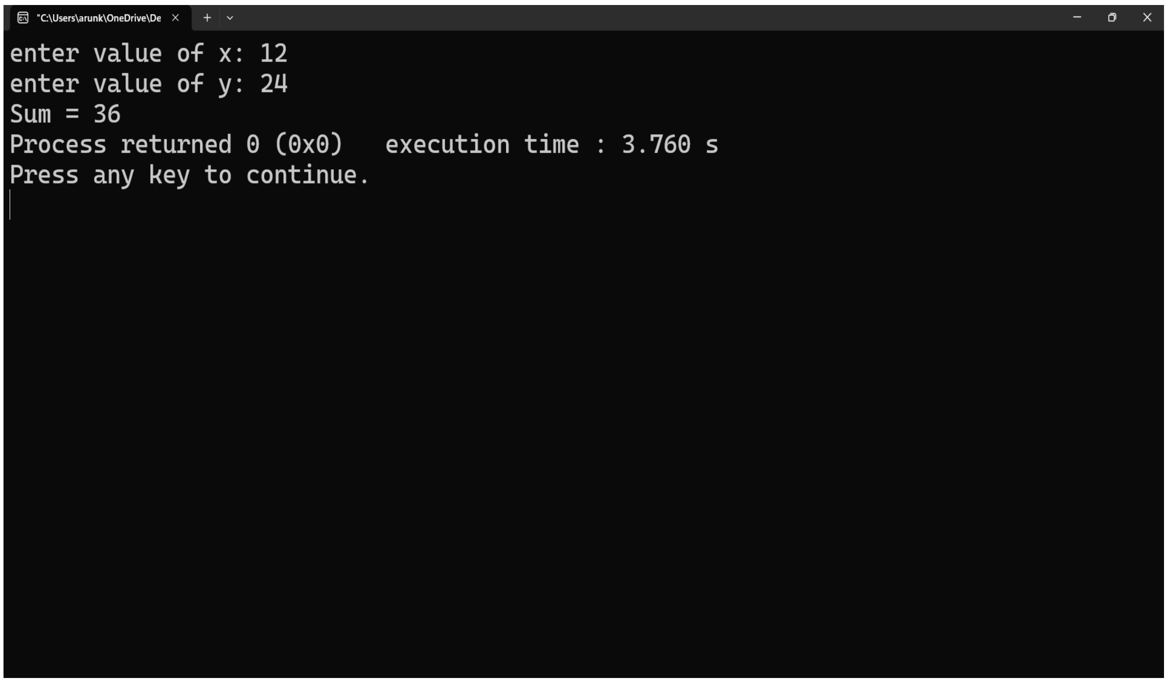
13. Write a C++ Program to multiply the values using multiple inheritance.

```
#include <iostream>
using namespace std;
```

```
class A
{
        public:
        int x;
        void getx()
   {
           cout << "enter value of x: "; cin >> x;
   }
};
class B
{
        public:
        int y;
        void gety()
        {
           cout << "enter value of y: "; cin >> y;
        }
};
class C : public A, public B
{
        public:
        void sum()
        {
           cout << "Sum = " << x + y;
        }
};

int main()
{
         C obj1;
         obj1.getx();
         obj1.gety();
         obj1.sum();
         return 0;
}
```

Output:

EX.No: 14	**Write a C++ Program to display the address values using pointer**
DATE:	



```
enter value of x: 12
enter value of y: 24
Sum = 36
Process returned 0 (0x0)   execution time : 3.760 s
Press any key to continue.
```

<u>RESULT:</u>

Thus the Above Program was Executed Successfully.

Aim:

To write a C++ program that demonstrates how to display the address of variables using pointers.

Algorithm:

Step 1: Start the Process.

Step 2: Declare variables of different data types (e.g., integer, float).

Step 3: Declare pointer variables corresponding to each of these data types.

Step 4: Assign the address of each variable to its corresponding pointer using the address-of operator (&).

Step 5: Use the pointers to display the memory addresses of the variables.

Step 6: Stop the process.

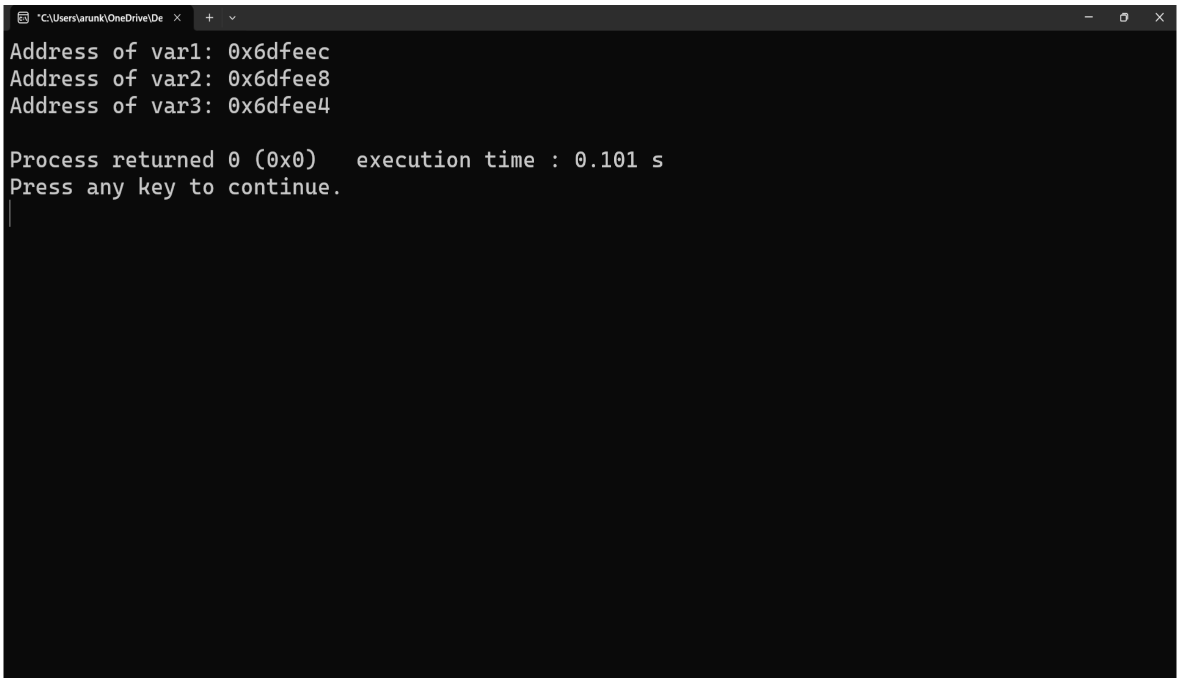
14. Write a C++ Program to display the address values using pointer

```
#include <iostream>
```

```
using namespace std;


int main()

{



   int var1 = 3;

   int var2 = 24;

   int var3 = 17;

   cout << "Address of var1: "<< &var1 << endl;

   cout << "Address of var2: " << &var2 << endl;

   cout << "Address of var3: " << &var3 << endl;

}
```

Output:

EX.No: 15	**Write a C++ program to demonstrate the use of class templates.**
DATE:	



```
Address of var1: 0x6dfeec
Address of var2: 0x6dfee8
Address of var3: 0x6dfee4

Process returned 0 (0x0)   execution time : 0.101 s
Press any key to continue.
```

<u>**RESULT:**</u>

Thus the Above Program was Executed Successfully.

<u>**Aim:**</u>

To write a C++ program that demonstrates the use of class templates. The program will create a generic class that can handle different data types using templates.

Algorithm:

Step 1: Start the process.

Step 2: Define a class template using the template keyword to allow the class to accept different data types.

Step 3: Declare a class Calculator with member functions to perform basic arithmetic operations (addition, subtraction, multiplication, division).

Step 4: Implement the member functions to handle generic data types (e.g., int, float, double).

Step 5: In the main() function:

• Create objects of the Calculator class with different data types (e.g., int, float).

• Perform arithmetic operations using the objects.

Step 6: Stop the process.

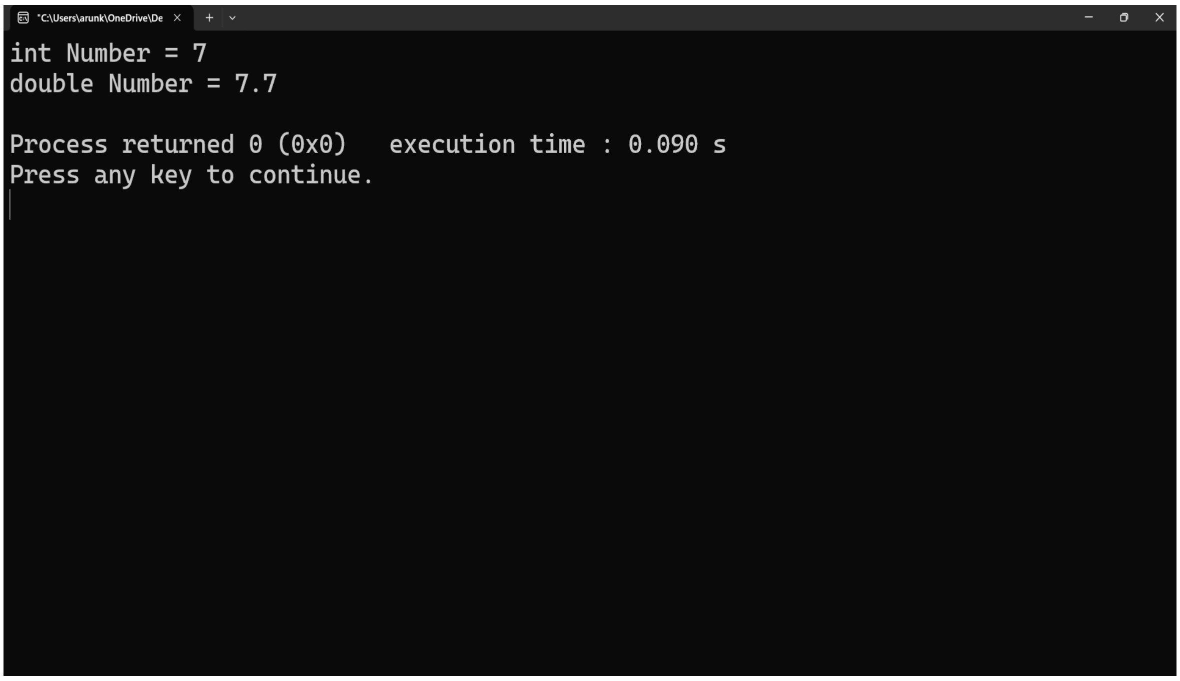
15. Write a C++ program to demonstrate the use of class templates.

```
#include <iostream>
using namespace std;
```

```
template <class T>
class Number
{
 private:
 T num;
 public:
 Number(T n) : num(n)
 {
 }
 T getNum()
 {
 return num;
 }
};
int main()
{
 Number<int> numberInt(7);
 Number<double> numberDouble(7.7);
 cout << "int Number = " << numberInt.getNum() << endl;
 cout << "double Number = " << numberDouble.getNum() << endl;
 return 0;
}
```

Output:

EX.No: 16	**Write a C++ Program to display the student ID and Name using Arrays within a class.**
DATE:	



```
int Number = 7
double Number = 7.7

Process returned 0 (0x0)   execution time : 0.090 s
Press any key to continue.
```

RESULT:

Thus the Above Program was Executed Successfully.

Aim:

To write a C++ program that displays student ID and name using arrays within a class. The program will demonstrate how to use arrays to store and display multiple student details.

Algorithm:

Step 1: Start the process.

Step 2: Define a class Student that will have arrays to store multiple students' names and IDs.

Step 3: Declare member functions in the Student class:

Step 4: A function to input student details using arrays.

Step 5: A function to display student details using arrays.

Step 6: In the main() function:

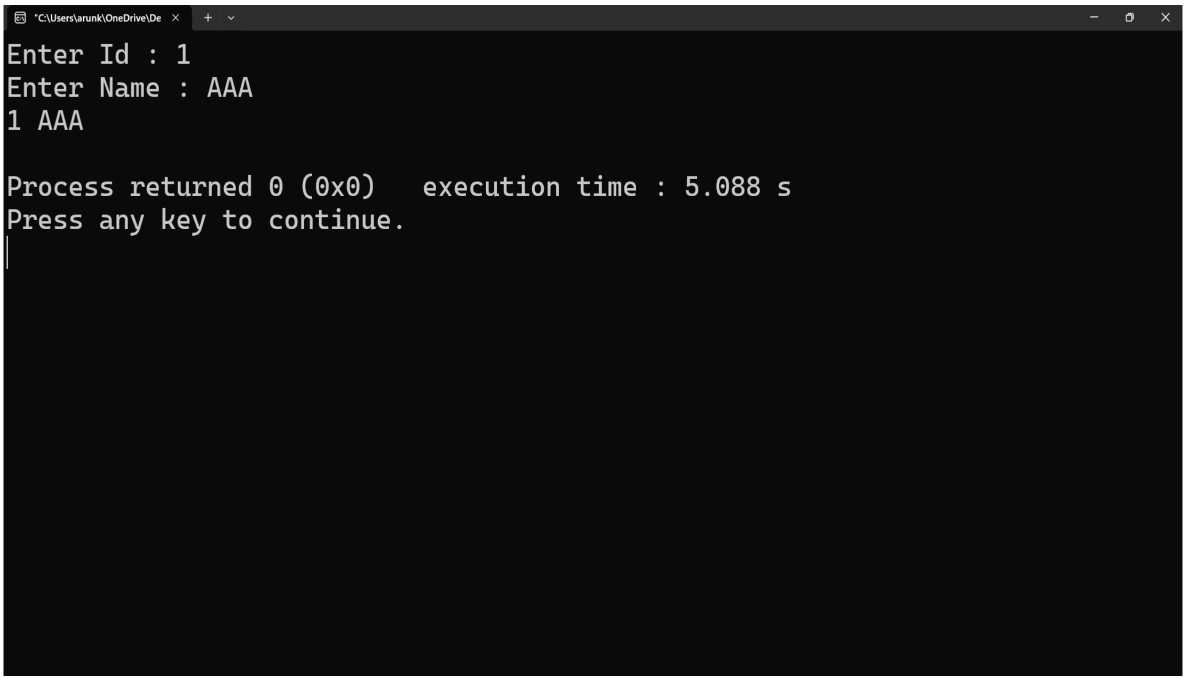
- Create an object of the Student class.
- Use the object to call the member functions to input and display the student details.

Step 7: Stop the process.

16.Write a C++ Program to multiply the values using multiple inheritance

```
#include<iostream>
using namespace std;

class Employee
{
 int id;
 char name[30];
 public:
 void getdata();
 void putdata();
};
void Employee::getdata()
{
 cout<<"Enter Id : ";
 cin>>id;
 cout<<"Enter Name : ";
 cin>>name;
}
void Employee::putdata()
{
 cout<<id<<" ";
 cout<<name<<" ";
 cout<<endl;
}
int main()
{
 Employee emp; //One member
 emp.getdata();//Accessing the function
 emp.putdata();//Accessing the function
 return 0;
}
```

Output:



```
Enter Id : 1
Enter Name : AAA
1 AAA

Process returned 0 (0x0)   execution time : 5.088 s
Press any key to continue.
```

RESULT:

Thus the Above Program was Executed Successfully.