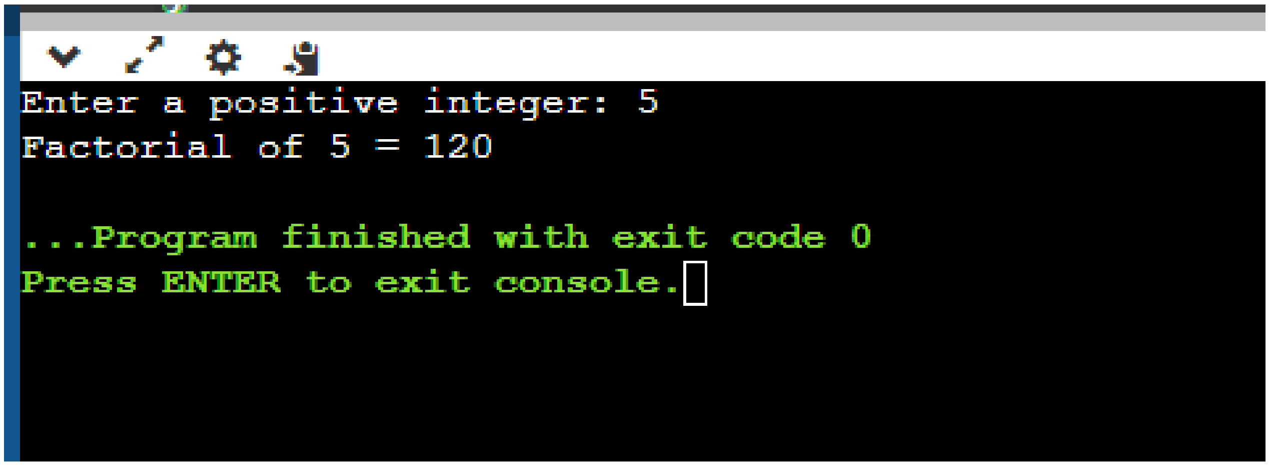
Q.13: Wap to print factorial.

<u>Code:</u>

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

int main() {
   int n;
   long factorial = 1.0;

   cout << "Enter a positive integer: ";
   cin >> n;

   if (n < 0)
      cout << "Error! Factorial of a negative number doesn't exist.";
   else {
      for(int i = 1; i <= n; ++i) {
         factorial *= i;
      }
      cout << "Factorial of " << n << " = " << factorial;
   }

   return 0;
}
```

Output:



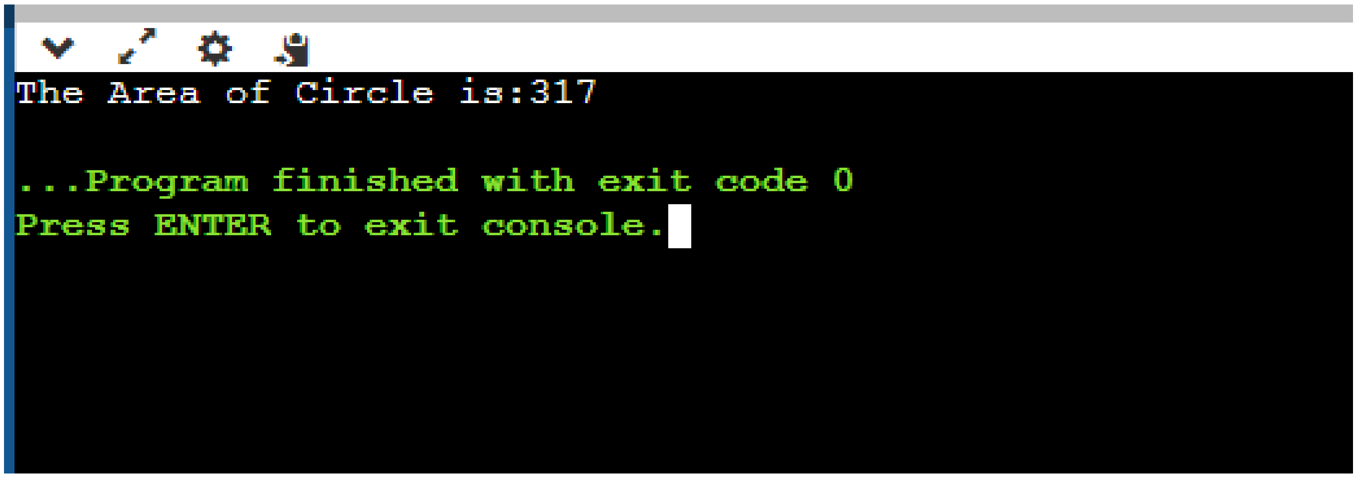
```
Enter a positive integer: 5
Factorial of 5 = 120

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

Q.14: Define a class in C++ and provide an example.

<u>Code:</u>

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

class Circle{

  private:
    float radious;
    float area;

  public:
    void setRadious(float radious){
        this->radious=radious;
    }

    float getArea(){
        area=3.17*radious*radious;
        return area;
    }

};

int main(){
    Circle c1;


    c1.setRadious(10);

    cout<<"The Area of Circle is:"<<c1.getArea();




    return 0;

}
```

Output:



```
The Area of Circle is:317

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

Q.15: Differentiate between a class and an object in C++.

Answer:

Classes are used to define the structure and behavior of objects, while objects . Classes are typically defined once and can be used to create multiple objects, while each is a unique class instance. A class is a blueprint for declaring and creating objects. An object is a class instance that allows programmers to use variables and methods from inside the class. Memory is not allocated to classes. Classes have no physical existence. When objects are created, memory is allocated to them in the heap memory.

Q.16: Discuss the concept of inheritance and its types in C++.

Answer:

Inheritance is a feature or a process in which, new classes are created from the existing classes. The new class created is called “derived class” or “child class” and the existing class is known as the “base class” or “parent class”. The derived class now is said to be inherited from the base class.

Types of Inheritance in C++

1. Single Inheritance: In single inheritance, a class is allowed to inherit from only one class. i.e. one subclass is inherited by one base class only.

2. Multiple Inheritance: Multiple Inheritance is a feature of C++ where a class can inherit from more than one class. i.e one subclass is inherited from more than one base class.

3. Multilevel Inheritance: In this type of inheritance, a derived class is created from another derived class.

4. Hierarchical Inheritance: In this type of inheritance, more than one subclass is inherited from a single base class. i.e. more than one derived class is created from a single base class.

5. Hybrid (Virtual) Inheritance: Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.

Q.17: What is a friend function, and how does it relate to OOP in C++?

Answer:

The friend function is a function that can access the private member of class, it’s written outside of class and declared in class. It will take an object of the class as a parameter.

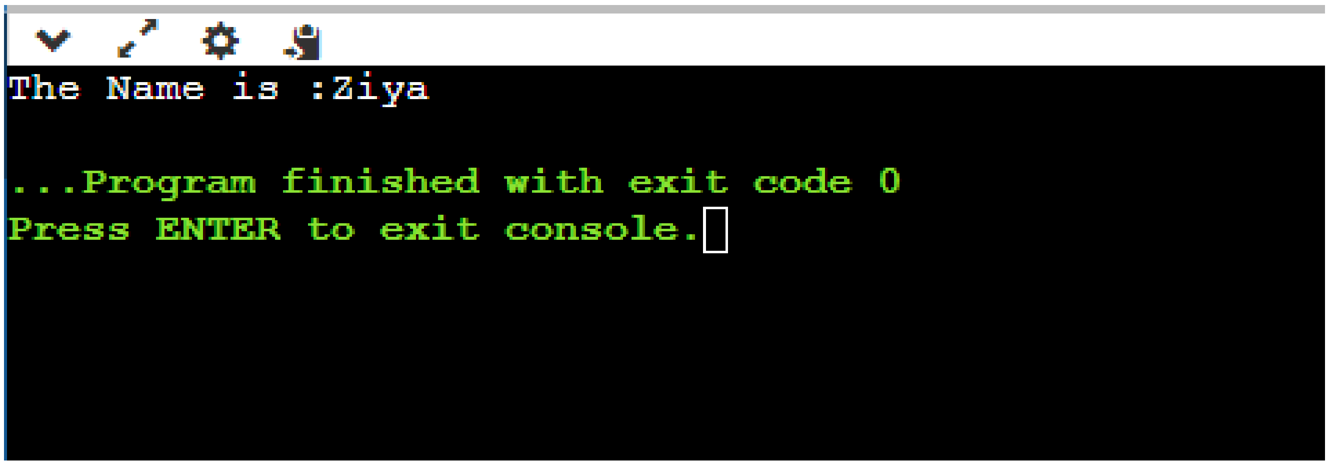
Object Oriented Programming has a future that is an abstraction where we can declare data members of the class as private, public, and protected for security purposes.

Code:

```
#include<iostream>

using namespace std;


class MCA{


  private:

     string name;


     friend void display(MCA s1);


  public:

    void setData(string name){

        this->name=name;

    }

};
```

```
void display(MCA s1){

    cout<<"The Name is :"<<s1.name;
}

int main(){

    MCA s1;
    s1.setData("Ziya");
    display(s1);

    return 0;
}
```

Output:



```
The Name is :Ziya

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

Q.18: What are virtual functions, and why are they essential in polymorphism with example.

Answer:

A C++ virtual function is a member function in the base class that you redefine in a derived class. It is declared using the virtual keyword. It is used to tell the compiler to perform dynamic linkage or late binding on the function.

There is a necessity to use the single pointer to refer to all the objects of the different classes. So, we create the pointer to the base class that refers to all the derived objects. But, when the base class pointer contains the address of the derived class object, always executes the base class function. This issue can only be resolved by using the 'virtual' function.

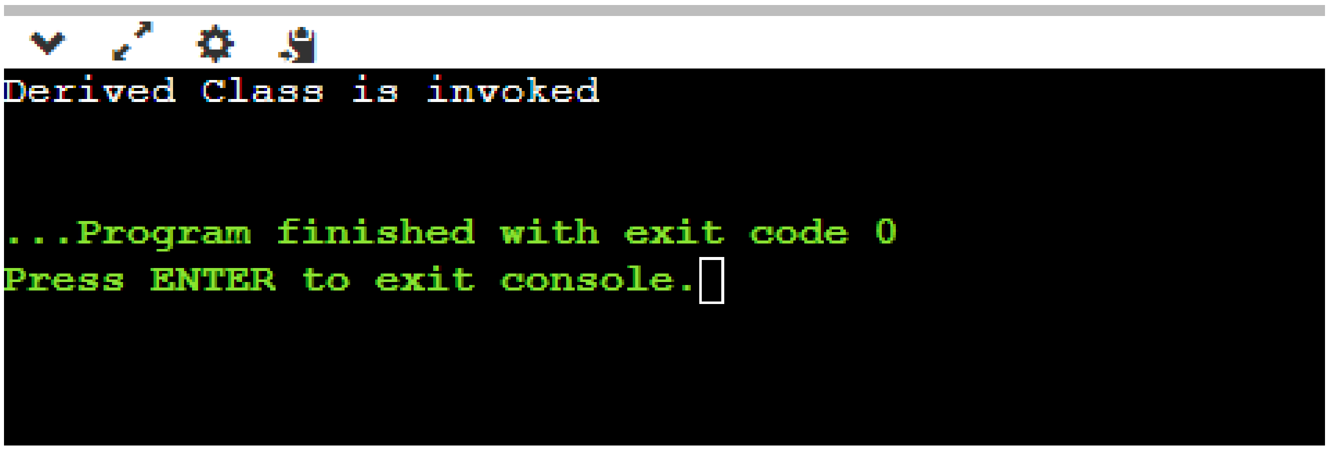
A 'virtual' is a keyword preceding the normal declaration of a function. When the function is made virtual, C++ determines which function is to be invoked at the runtime based on the type of the object pointed by the base class pointer.

Code:

```
#include <iostream>

using namespace std;


class A{

 public:

 virtual void display()

 {

  cout << "Base class is invoked"<<endl;

 }

};

class B:public A
```

```
{
 public:
 void display()
 {
  cout << "Derived Class is invoked"<<endl;
 }
};
int main()
{
 A* a;    //pointer of base class
 B b;     //object of derived class
 a = &b;
 a->display();   //Late Binding occurs
}
```

Output:



```
Derived Class is invoked


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

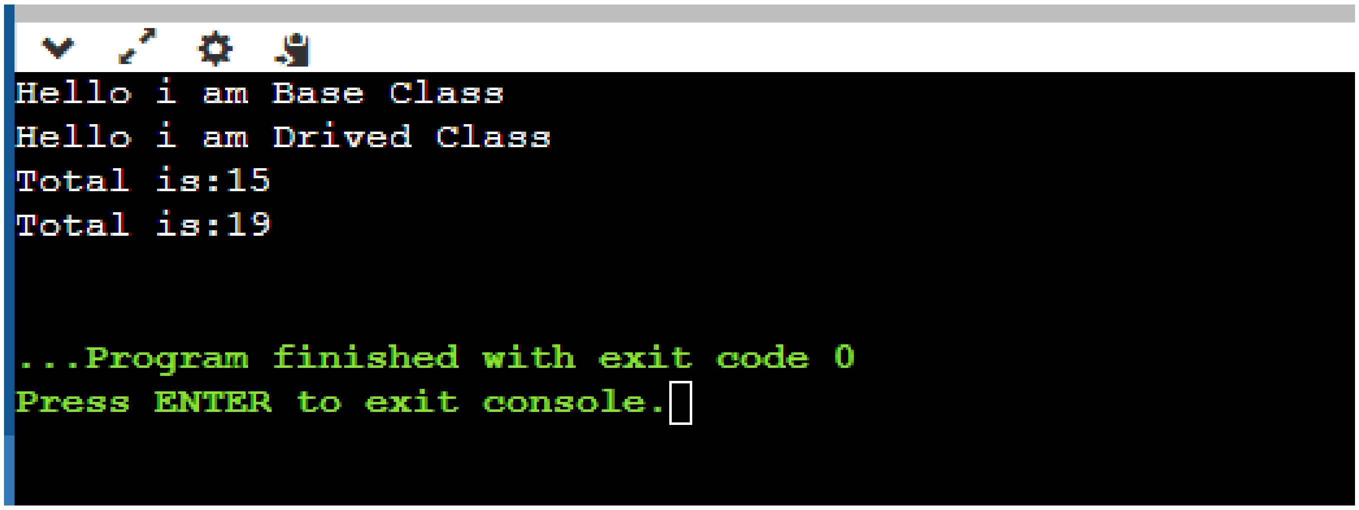
Q.19: WAP to use functions overloading and overriding.

<u>Code:</u>

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

class Base{

  public:
    void display(){
        cout<<"Hello i am Base Class"<<endl;
    }

};

class Drived : public Base{

  public:
    void display(){
        cout<<"Hello i am Drived Class"<<endl;
    }

    void add(int a,int b){
        cout<<"Total is:"<<a+b<<endl;
    }
```

```
     void add(int a,int b,int c){
         cout<<"Total is:"<<a+b+c<<endl;
     }
};



int main(){


    Base b1;
    b1.display();


    Drived d1;
    d1.display();


    d1.add(1,5,9);
    d1.add(14,5);



    return 0;


}
```

Output:



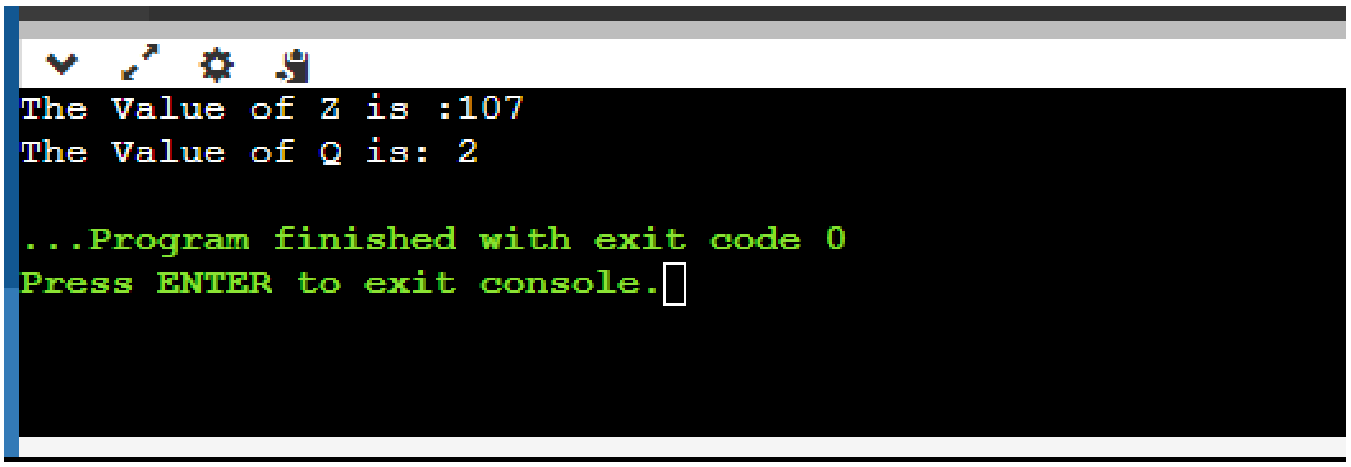
```
Hello i am Base Class
Hello i am Drived Class
Total is:15
Total is:19


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

Q.20: WAP to perform implicit and explicit type casting .

Code:

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

int main(){

    int x=10;
    char y='a';

    int z=x + y;

    cout<<"The Value of Z is :"<<z<<endl;

    double p=1.2;

    int q=(int)p+1;
    cout<<"The Value of Q is: "<<q;

    return 0;
}
```

Output:



```
The Value of Z is :107
The Value of Q is: 2

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

Q.21: Explain the concept of encapsulation and how it is implemented in C++.

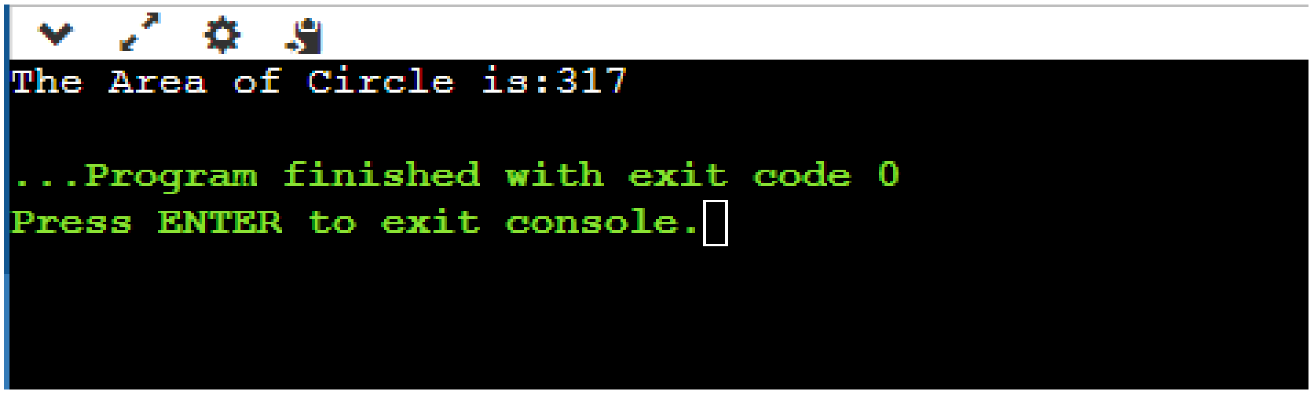
Answer:

Encapsulation is the feature of OOPs that helps to bind the data in a single unit or we can say grouping of method and data in a single unit. for example, we have an employee and they have a method to set the personal details of the employee another one is to get personnel to extract the personal details of an employee when we wrap in a single unit it's called encapsulation. it provides the security of data.

Code:

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

class Circle{

  private:
    float radious;
    float area;

  public:
    void setRadious(float radious){
        this->radious=radious;
    }

    float getArea(){
        area=3.17*radious*radious;
        return area;
    }

};

int main(){

    Circle c1;

    c1.setRadious(10);
    cout<<"The Area of Circle is:"<<c1.getArea();




    return 0;
}
```

Output:



```
The Area of Circle is:317

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

Q.22: How does polymorphism enhance the flexibility of C++ programs?

Answer:

Polymorphism promotes code flexibility by allowing objects of different types to be treated as objects of a common type. Polymorphism is a fundamental concept in object-oriented programming (OOP) that provides the ability to use an entity in multiple forms. It allows objects of different types to be treated as objects of a common type, which leads to a more flexible and easily managed code.