Objective:

- To understand the concept of **class** as a blueprint for creating objects in C++.
- To learn how to define a class with data members and member functions.
- To create and use **objects** of a class for storing data and performing operations.
- To implement **encapsulation** by combining data and functions in a single unit.
- To practice basic **object-oriented programming** principles like abstraction and data hiding.

Class and Object in C++

Theory:

- A **class** is a user-defined data type that acts as a blueprint for creating objects. It encapsulates data (attributes) and functions (methods) that operate on the data.
- An **object** is an instance of a class. When a class is defined, no memory is allocated, but when an object is created, memory is allocated for the object.
- Classes support the concept of **Encapsulation**, which means combining data and functions that manipulate that data into a single unit.
- Using classes and objects helps implement **Object-Oriented Programming (OOP)** principles such as abstraction, encapsulation, inheritance, and polymorphism.

Syntax:

```
// Class definition

class ClassName {

   // Access specifier

   public:

      // Data members (attributes)

      dataType variable1;

      dataType variable2;

      // Member functions (methods)
```

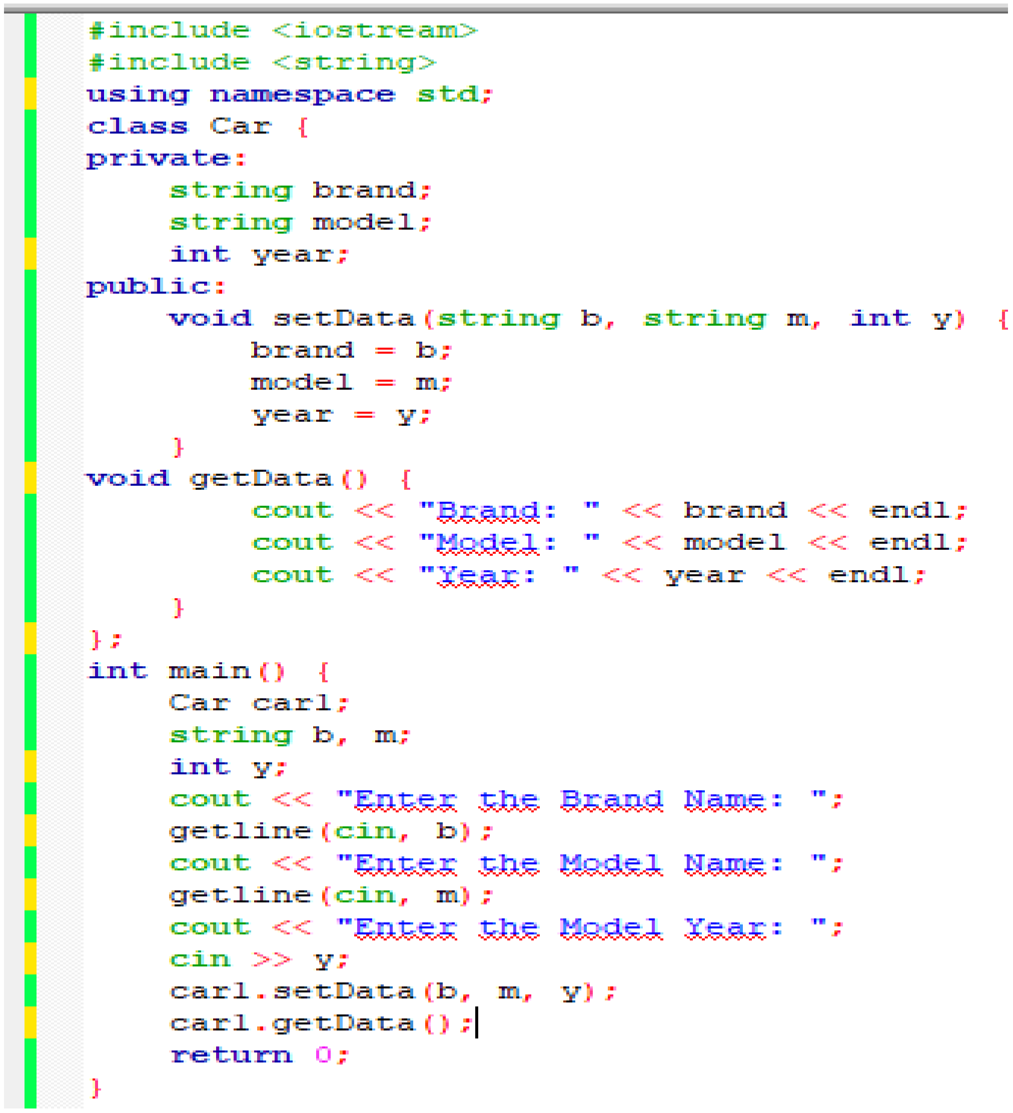
```
        void functionName() {
            // function body
        }
};
// Creating an object of the class
ClassName objectName;
```

Example:

```
#include <iostream>
using namespace std;
// Class definition
class Car {
public:
    string brand;
    int year;
    void display() {
        cout << "Brand: " << brand << endl;
        cout << "Year: " << year << endl;}};
int main() {
    // Creating an object of class Car
    Car myCar;
    // Accessing data members
    myCar.brand = "Toyota";
    myCar.year = 2020;
    // Calling member function
    myCar.display();
    return 0;
}
```

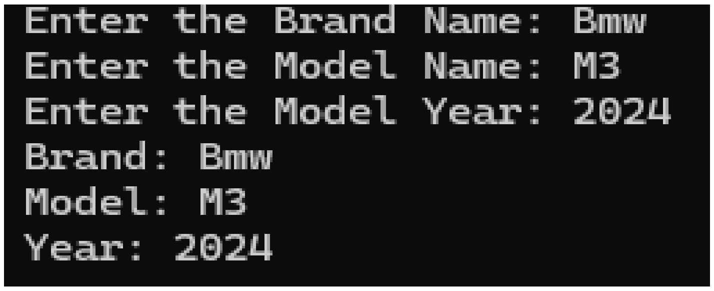
Qn 1: Define a class Car with private members brand, model, and year. Include public member functions to set and get these private members. Ensure that only member functions can access these private members.

Code:



```
#include <iostream>
#include <string>
using namespace std;
class Car {
private:
    string brand;
    string model;
    int year;
public:
    void setData(string b, string m, int y) {
        brand = b;
        model = m;
        year = y;
    }
void getData() {
        cout << "Brand: " << brand << endl;
        cout << "Model: " << model << endl;
        cout << "Year: " << year << endl;
    }
};
int main() {
    Car car1;
    string b, m;
    int y;
    cout << "Enter the Brand Name: ";
    getline(cin, b);
    cout << "Enter the Model Name: ";
    getline(cin, m);
    cout << "Enter the Model Year: ";
    cin >> y;
    car1.setData(b, m, y);
    car1.getData();
    return 0;
}
```

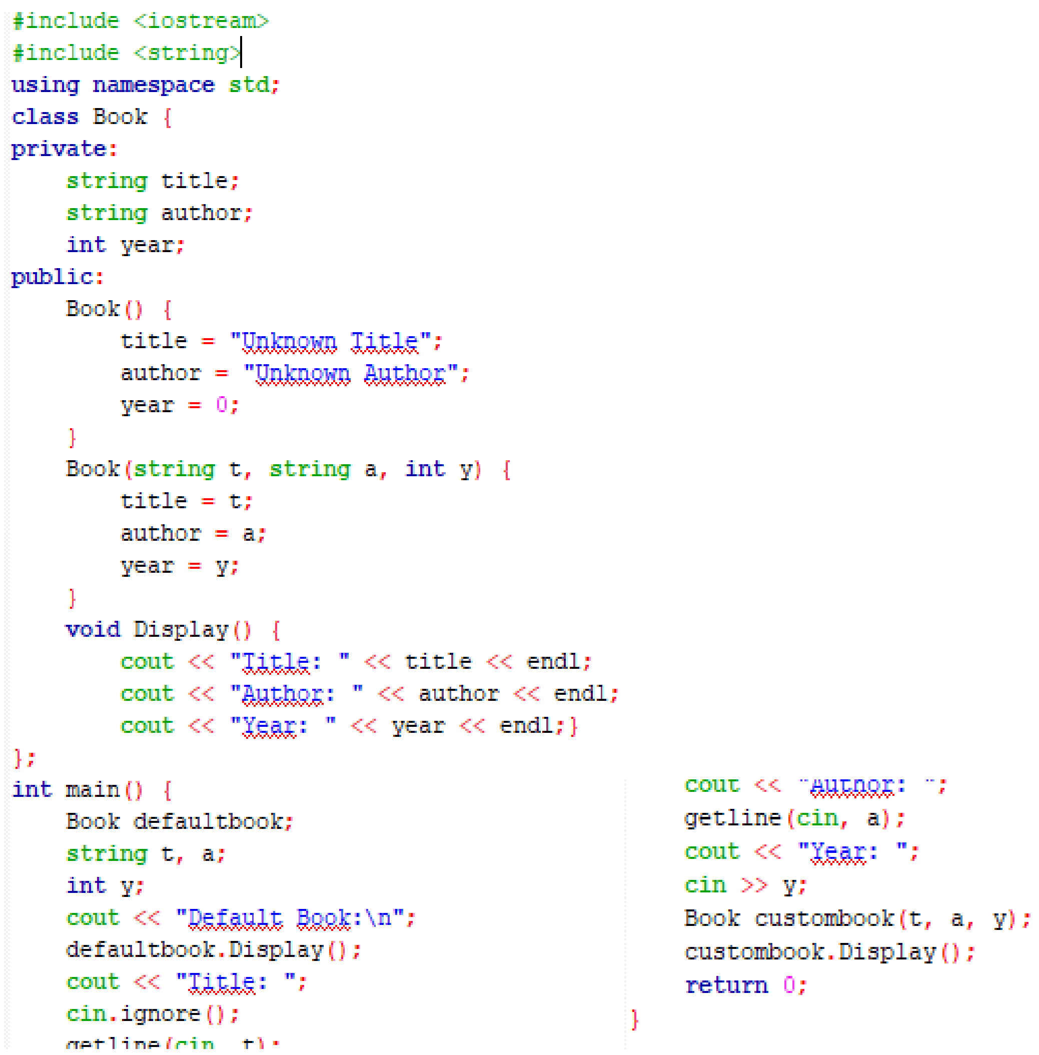
Output:



```
Enter the Brand Name: Bmw
Enter the Model Name: M3
Enter the Model Year: 2024
Brand: Bmw
Model: M3
Year: 2024
```

Qn 2: Define a class Book with private members title, author, and year. Implement both default and parameterized constructors. The default constructor should initialize the members with default values, and the parameterized constructor should set these values based on user input. Provide a method to display the details of a book.

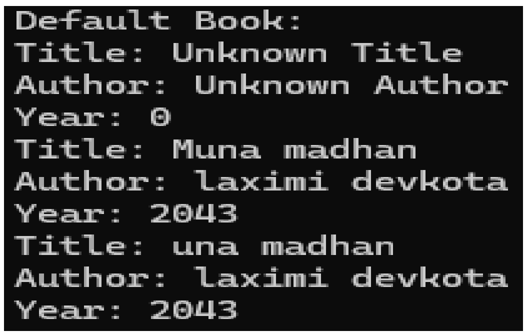
Code:



```cpp
#include <iostream>
#include <string>
using namespace std;
class Book {
private:
    string title;
    string author;
    int year;
public:
    Book() {
        title = "Unknown Title";
        author = "Unknown Author";
        year = 0;
    }
    Book(string t, string a, int y) {
        title = t;
        author = a;
        year = y;
    }
    void Display() {
        cout << "Title: " << title << endl;
        cout << "Author: " << author << endl;
        cout << "Year: " << year << endl;}
};
int main() {
    Book defaultbook;
    string t, a;
    int y;
    cout << "Default Book:\n";
    defaultbook.Display();
    cout << "Title: ";
    cin.ignore();
    getline(cin, t);
```

```cpp
    cout << "Author: ";
    getline(cin, a);
    cout << "Year: ";
    cin >> y;
    Book custombook(t, a, y);
    custombook.Display();
    return 0;
}
```

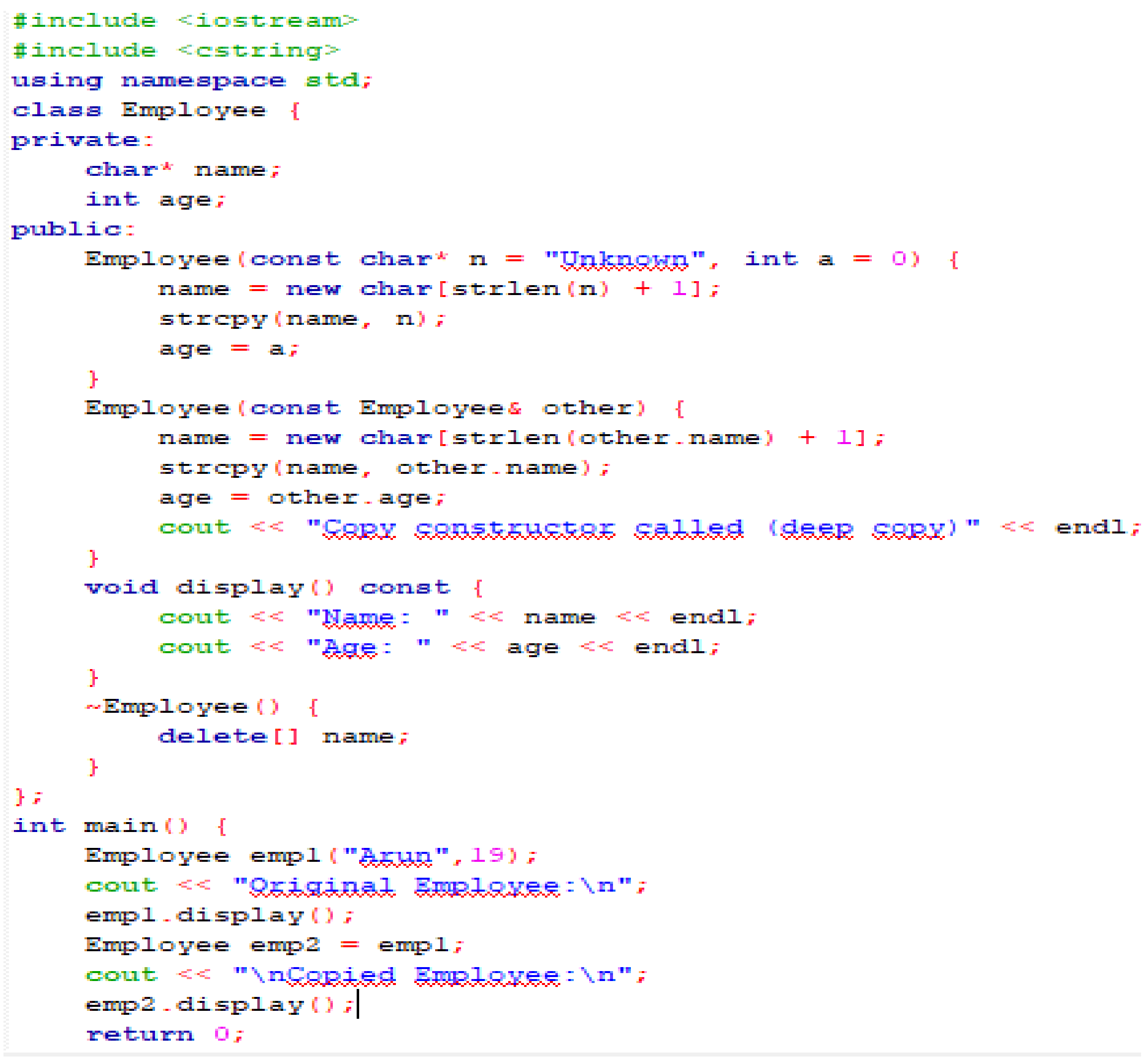
Output:



```
Default Book:
Title: Unknown Title
Author: Unknown Author
Year: 0
Title: Muna madhan
Author: laximi devkota
Year: 2043
Title: una madhan
Author: laximi devkota
Year: 2043
```

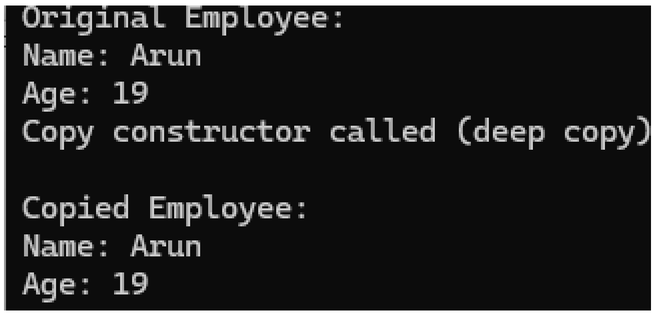
Qn 3: Create a class Employee with private members name and age. Implement a copy constructor to create a deep copy of an existing Employee object. Provide a method to display the details of an employee.

Code:



```
#include <iostream>
#include <cstring>
using namespace std;
class Employee {
private:
    char* name;
    int age;
public:
    Employee(const char* n = "Unknown", int a = 0) {
        name = new char[strlen(n) + 1];
        strcpy(name, n);
        age = a;
    }
    Employee(const Employee& other) {
        name = new char[strlen(other.name) + 1];
        strcpy(name, other.name);
        age = other.age;
        cout << "Copy constructor called (deep copy)" << endl;
    }
    void display() const {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
    ~Employee() {
        delete[] name;
    }
};
int main() {
    Employee emp1("Arun",19);
    cout << "Original Employee:\n";
    emp1.display();
    Employee emp2 = emp1;
    cout << "\nCopied Employee:\n";
    emp2.display();
    return 0;
```

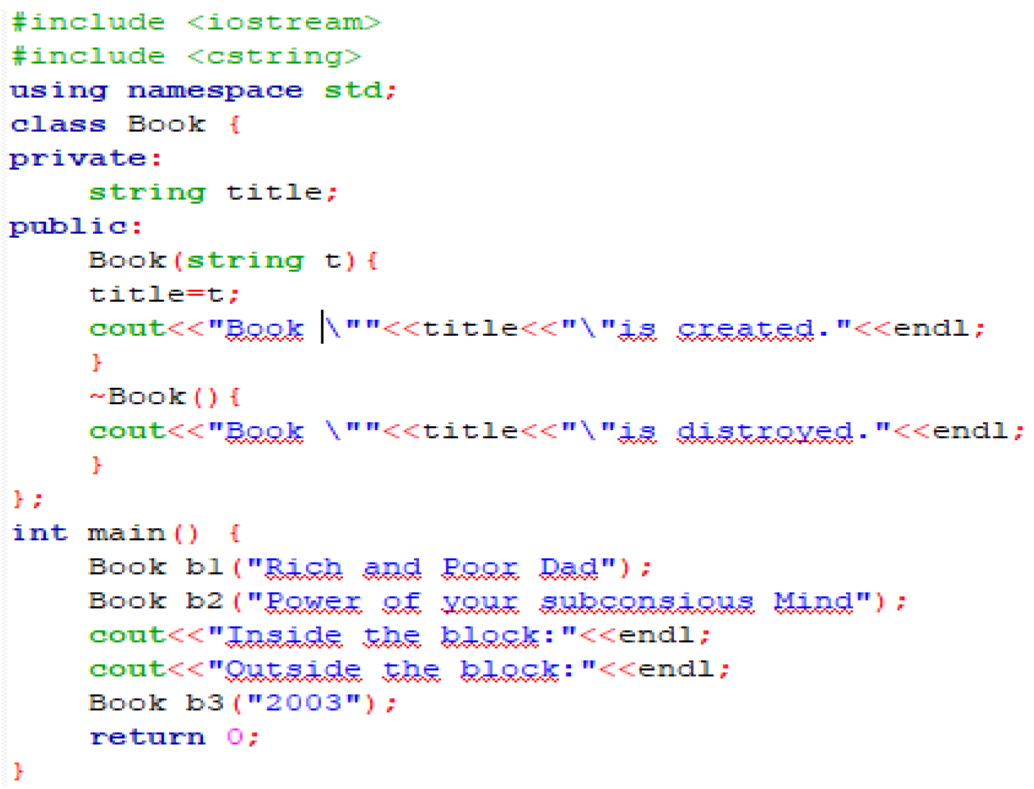
Ouput:



```
Original Employee:
Name: Arun
Age: 19
Copy constructor called (deep copy)

Copied Employee:
Name: Arun
Age: 19
```

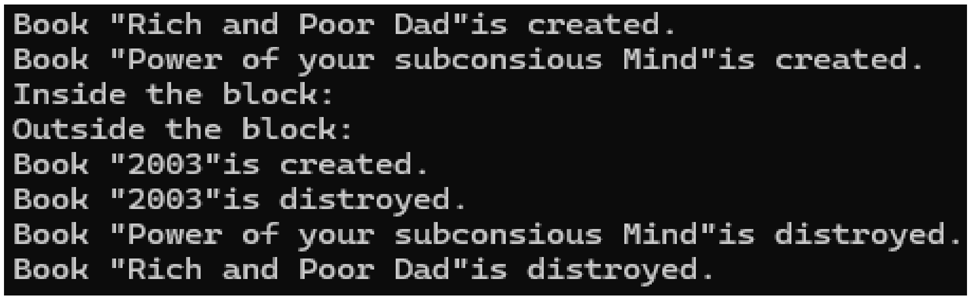
Qn 4: Define a class Book with a private member title. Implement a constructor that initializes title and a destructor that prints a message when the Book object is destroyed. Create instances of Book objects in main() to demonstrate the usage of the destructor.

Code:



```cpp
#include <iostream>
#include <cstring>
using namespace std;
class Book {
private:
	string title;
public:
	Book(string t){
	title=t;
	cout<<"Book \""<<title<<"\"is created."<<endl;
	}
	~Book(){
	cout<<"Book \""<<title<<"\"is distroyed."<<endl;
	}
};
int main()  {
	Book b1("Rich and Poor Dad");
	Book b2("Power of your subconsious Mind");
	cout<<"Inside the block:"<<endl;
	cout<<"Outside the block:"<<endl;
	Book b3("2003");
	return 0;
}
```

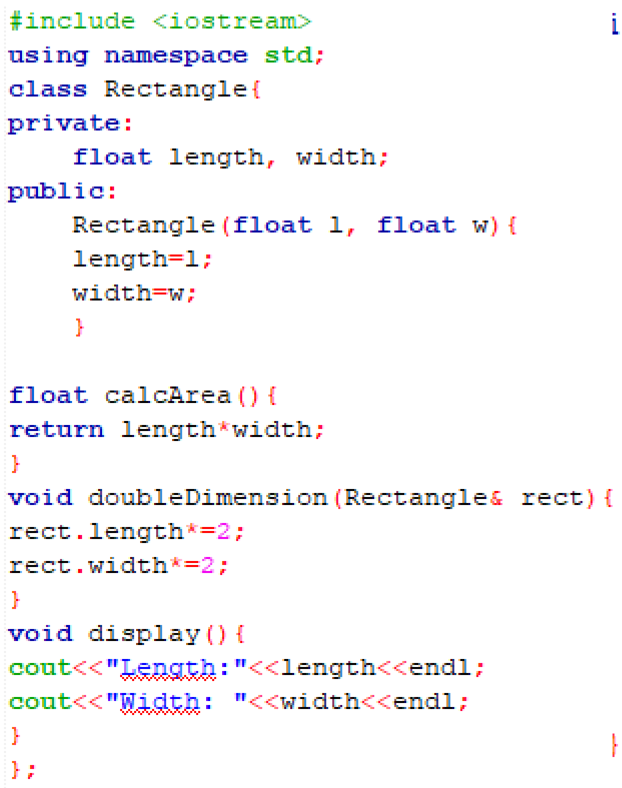
Output:



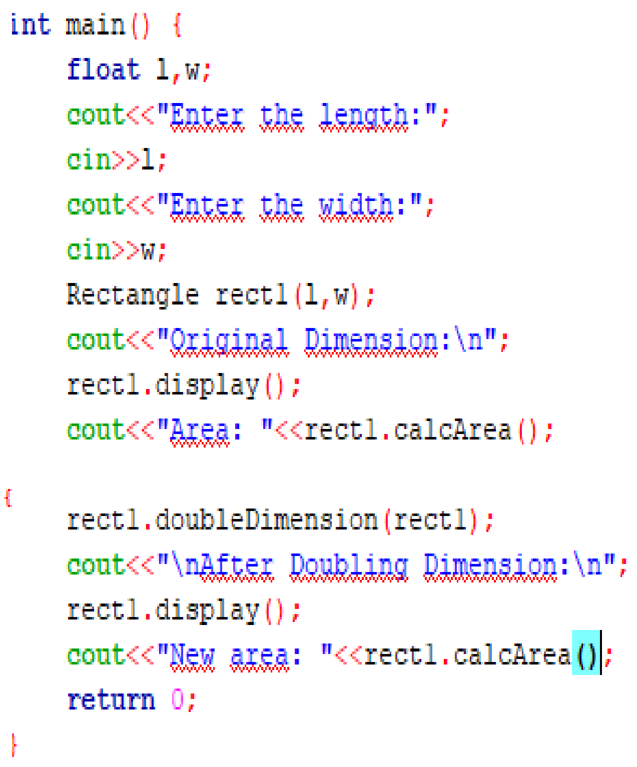
```
Book "Rich and Poor Dad"is created.
Book "Power of your subconsious Mind"is created.
Inside the block:
Outside the block:
Book "2003"is created.
Book "2003"is distroyed.
Book "Power of your subconsious Mind"is distroyed.
Book "Rich and Poor Dad"is distroyed.
```

Qn 5: Define a class Rectangle with private members length and width. Implement a constructor to initialize these members. Write a function calculateArea() that calculates and returns the area of the rectangle. Define another function doubleDimensions(Rectangle rect) that takes a Rectangle object as an argument and doubles its length and width. Demonstrate the usage of both functions in main().

Code:

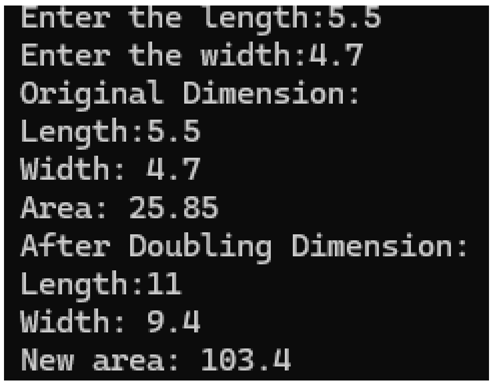


```
#include <iostream>
using namespace std;
class Rectangle{
private:
    float length, width;
public:
    Rectangle(float l, float w){
    length=l;
    width=w;
    }

float calcArea(){
return length*width;
}
void doubleDimension(Rectangle& rect){
rect.length*=2;
rect.width*=2;
}
void display(){
cout<<"Length:"<<length<<endl;
cout<<"Width: "<<width<<endl;
}
};
```



```
int main() {
    float l,w;
    cout<<"Enter the length:";
    cin>>l;
    cout<<"Enter the width:";
    cin>>w;
    Rectangle rect1(l,w);
    cout<<"Original Dimension:\n";
    rect1.display();
    cout<<"Area: "<<rect1.calcArea();

    rect1.doubleDimension(rect1);
    cout<<"\nAfter Doubling Dimension:\n";
    rect1.display();
    cout<<"New area: "<<rect1.calcArea();
    return 0;
}
```

Output:

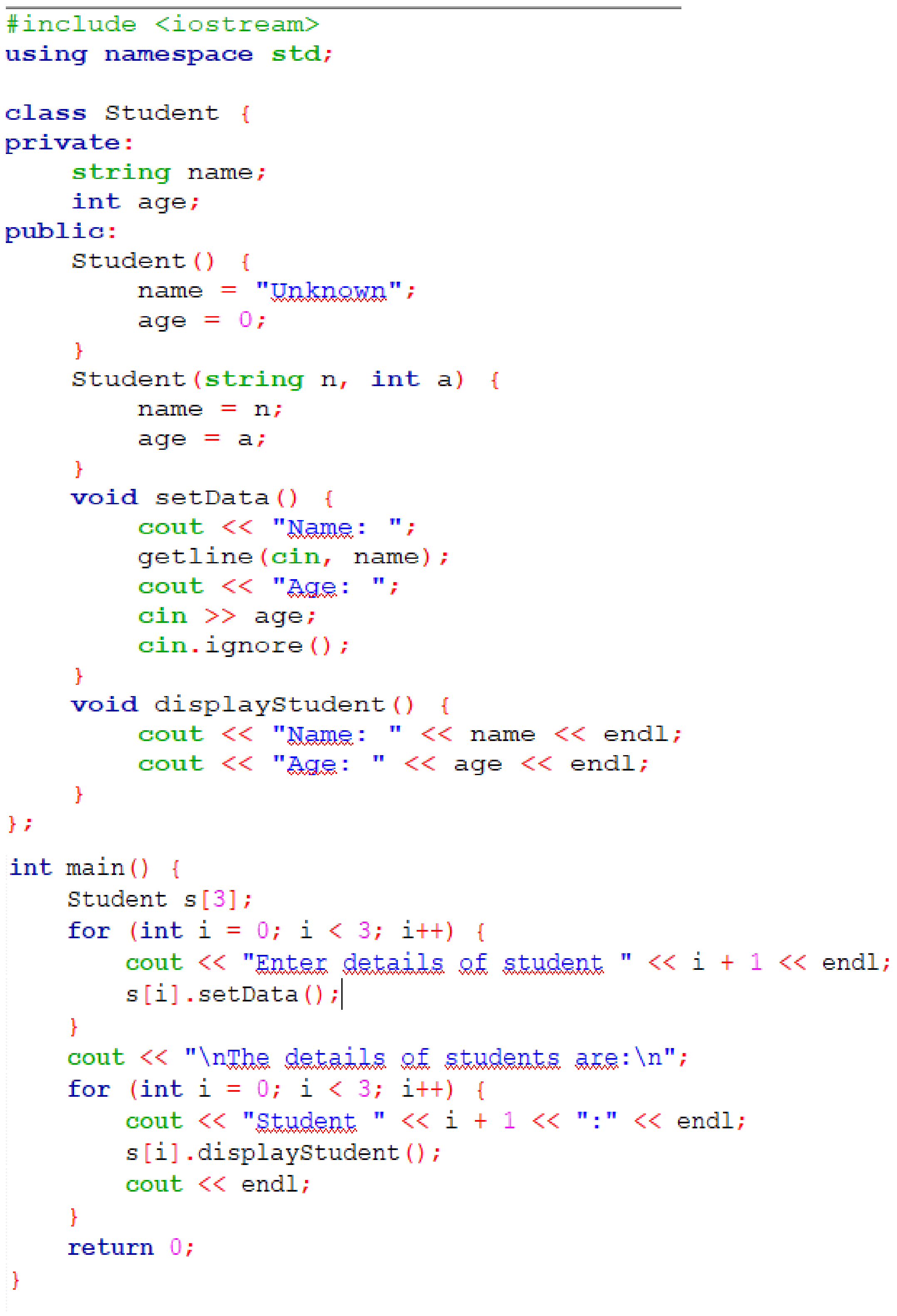


```
Enter the length:5.5
Enter the width:4.7
Original Dimension:
Length:5.5
Width: 4.7
Area: 25.85
After Doubling Dimension:
Length:11
Width: 9.4
New area: 103.4
```

Qn 6: Define a class Student with private members name and age. Implement a constructor to initialize these members. Create an array studentArray of size 3 to store objects of type Student.

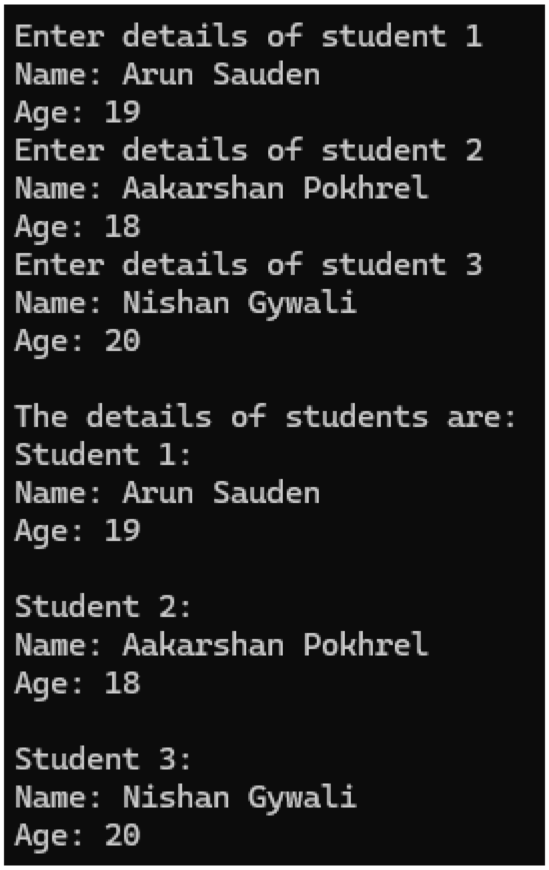
Populate the array with student details and display the details using a method displayStudentDetails().

Code:



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

class Student {
private:
    string name;
    int age;
public:
    Student() {
        name = "Unknown";
        age = 0;
    }
    Student(string n, int a) {
        name = n;
        age = a;
    }
    void setData() {
        cout << "Name: ";
        getline(cin, name);
        cout << "Age: ";
        cin >> age;
        cin.ignore();
    }
    void displayStudent() {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
};

int main() {
    Student s[3];
    for (int i = 0; i < 3; i++) {
        cout << "Enter details of student " << i + 1 << endl;
        s[i].setData();
    }
    cout << "\nThe details of students are:\n";
    for (int i = 0; i < 3; i++) {
        cout << "Student " << i + 1 << ":" << endl;
        s[i].displayStudent();
        cout << endl;
    }
    return 0;
}
```

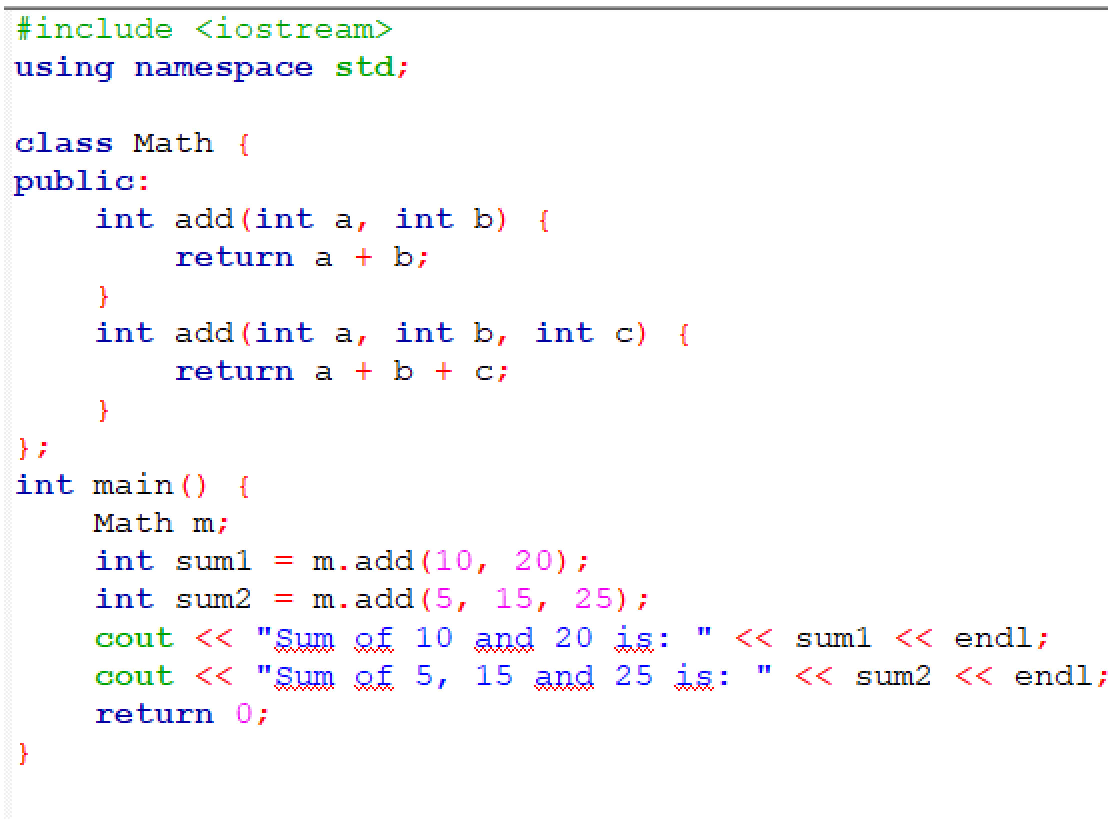
Output:



```
Enter details of student 1
Name: Arun Sauden
Age: 19
Enter details of student 2
Name: Aakarshan Pokhrel
Age: 18
Enter details of student 3
Name: Nishan Gywali
Age: 20

The details of students are:
Student 1:
Name: Arun Sauden
Age: 19

Student 2:
Name: Aakarshan Pokhrel
Age: 18

Student 3:
Name: Nishan Gywali
Age: 20
```

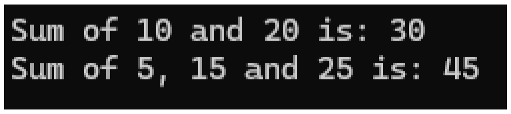
Qn 7: Define a class Math with two overloaded methods add(). The first method should take two integers as parameters and return their sum. The second method should take three integers as parameters and return their sum. Demonstrate the usage of both methods in main().

Code:



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

class Math {
public:
    int add(int a, int b) {
        return a + b;
    }
    int add(int a, int b, int c) {
        return a + b + c;
    }
};
int main() {
    Math m;
    int sum1 = m.add(10, 20);
    int sum2 = m.add(5, 15, 25);
    cout << "Sum of 10 and 20 is: " << sum1 << endl;
    cout << "Sum of 5, 15 and 25 is: " << sum2 << endl;
    return 0;
}
```

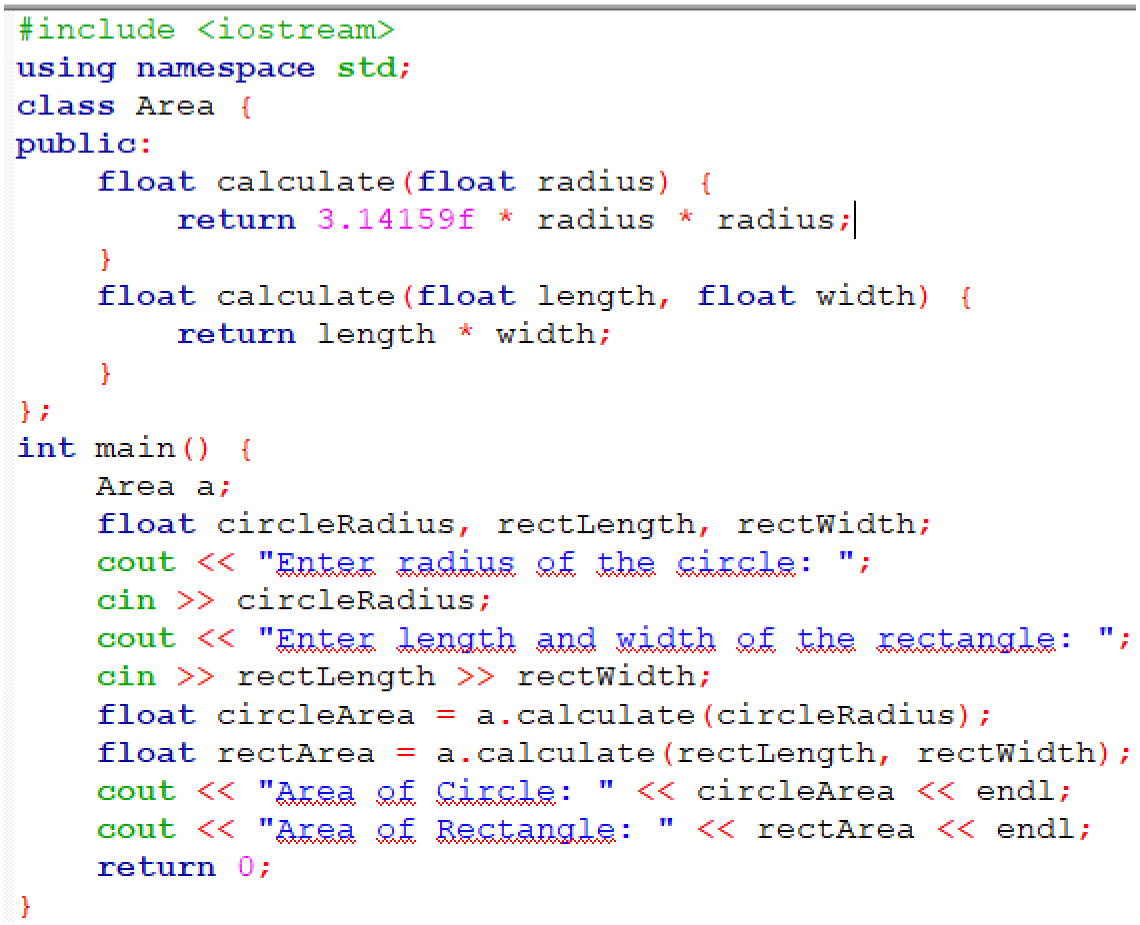
Output:



```
Sum of 10 and 20 is: 30
Sum of 5, 15 and 25 is: 45
```

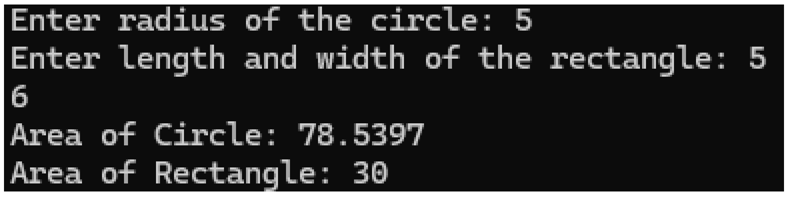
Qn 8: Implement a class Area with overloaded methods calculate(). The first method should take the radius of a circle as a parameter and return the area of the circle. The second method should take the length and width of a rectangle as parameters and return the area of the rectangle. Demonstrate the usage of both methods in main().

Code:



```cpp
#include <iostream>
using namespace std;
class Area {
public:
    float calculate(float radius) {
        return 3.14159f * radius * radius;
    }
    float calculate(float length, float width) {
        return length * width;
    }
};
int main() {
    Area a;
    float circleRadius, rectLength, rectWidth;
    cout << "Enter radius of the circle: ";
    cin >> circleRadius;
    cout << "Enter length and width of the rectangle: ";
    cin >> rectLength >> rectWidth;
    float circleArea = a.calculate(circleRadius);
    float rectArea = a.calculate(rectLength, rectWidth);
    cout << "Area of Circle: " << circleArea << endl;
    cout << "Area of Rectangle: " << rectArea << endl;
    return 0;
}
```

Output:



```
Enter radius of the circle: 5
Enter length and width of the rectangle: 5
6
Area of Circle: 78.5397
Area of Rectangle: 30
```

Discussion:

In this lab, we studied the basic concepts of class, object, constructor, and destructor in C++. A class acts as a template, while objects are real instances of that class. We used constructors to automatically initialize object data and destructors to release resources. This helped us understand how object lifecycle is managed in C++. The hands-on practice made it clear how object-oriented programming supports better code organization, memory handling, and reusability, which are essential in building efficient software applications.

Conclusion:

Through this lab, we gained a clear understanding of the core concepts of object-oriented programming, including class, object, constructor, and destructor. We learned how constructors simplify object initialization and how destructors help manage resources effectively. This experience strengthened our programming skills and highlighted the importance of structuring code using OOP principles. It provided a strong foundation for developing organized, reusable, and efficient C++ programs in future projects.