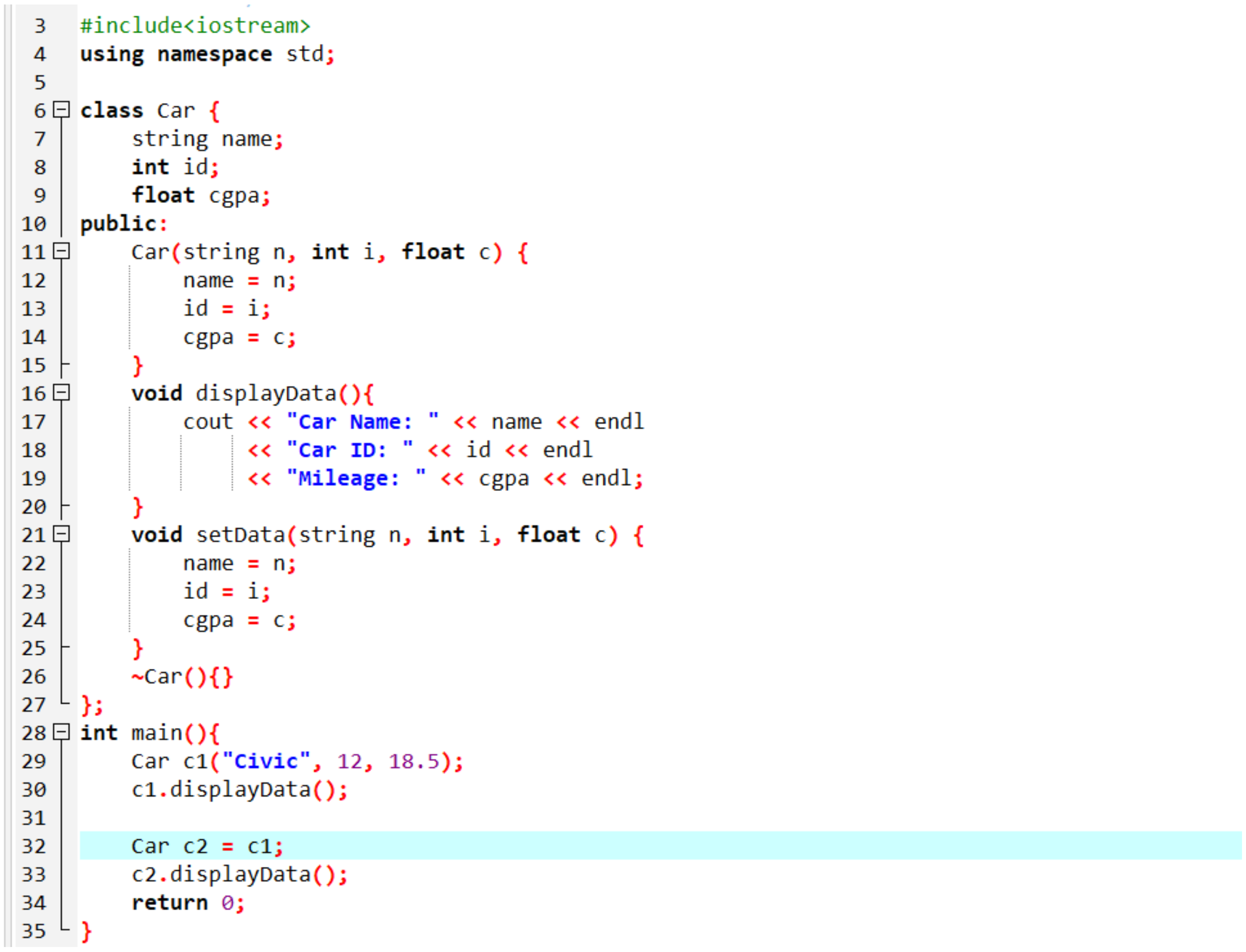
Areeba **CT-25070** **Section B**

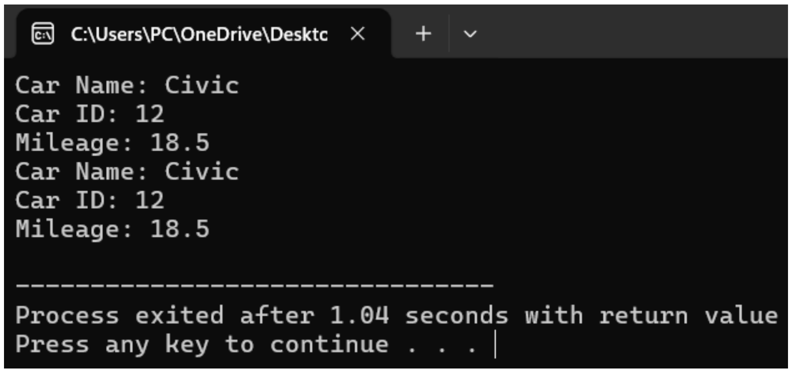
OBJECT ORIENTED PROGRAMMING (LAB 3)

EXERCISES

1. Write a C++ program to copy the value of one object to another object using copy constructor.



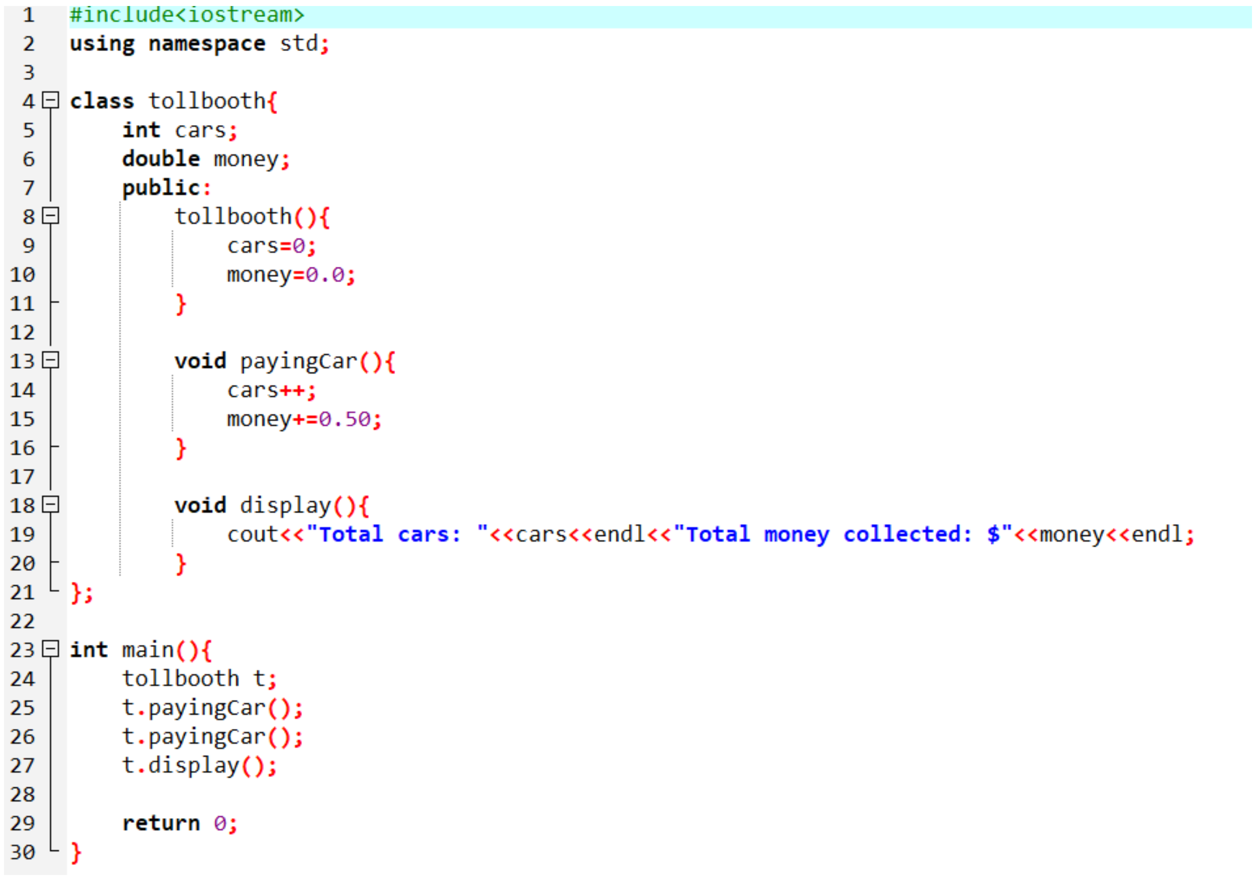
```cpp
#include<iostream>
using namespace std;

class Car {
    string name;
    int id;
    float cgpa;
public:
    Car(string n, int i, float c) {
        name = n;
        id = i;
        cgpa = c;
    }
    void displayData(){
        cout << "Car Name: " << name << endl
             << "Car ID: " << id << endl
             << "Mileage: " << cgpa << endl;
    }
    void setData(string n, int i, float c) {
        name = n;
        id = i;
        cgpa = c;
    }
    ~Car(){}
};
int main(){
    Car c1("Civic", 12, 18.5);
    c1.displayData();

    Car c2 = c1;
    c2.displayData();
    return 0;
}
```



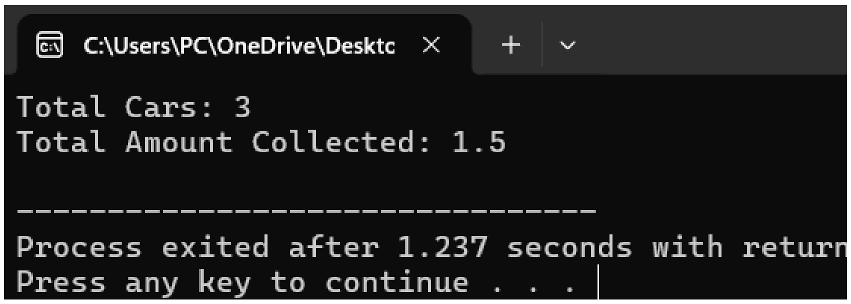
```
Car Name: Civic
Car ID: 12
Mileage: 18.5
Car Name: Civic
Car ID: 12
Mileage: 18.5

--------------------------------
Process exited after 1.04 seconds with return value
Press any key to continue . . .
```

2. Create a class tollbooth. The two data items are a type int to hold the total number of cars and a type double to hold the total amount of money collected. A constructor initializes both these to 0. When a car passes the toll, a member function called payingCar( ) increments the car total and adds 0.50 to the cash total. Another member function displays the two totals. DESIGN and IMPLEMENT:

this case. Make assumptions (if required) and include it in the description before designing the solution.



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

class tollbooth{
	int cars;
	double money;
	public:
		tollbooth(){
			cars=0;
			money=0.0;
		}
		
		void payingCar(){
			cars++;
			money+=0.50;
		}
		
		void display(){
			cout<<"Total cars: "<<cars<<endl<<"Total money collected: $"<<money<<endl;
		}
};

int main(){
	tollbooth t;
	t.payingCar();
	t.payingCar();
	t.display();
	
	return 0;
}
```



```
Total Cars: 3
Total Amount Collected: 1.5

--------------------------------
Process exited after 1.237 seconds with return
Press any key to continue . . .
```

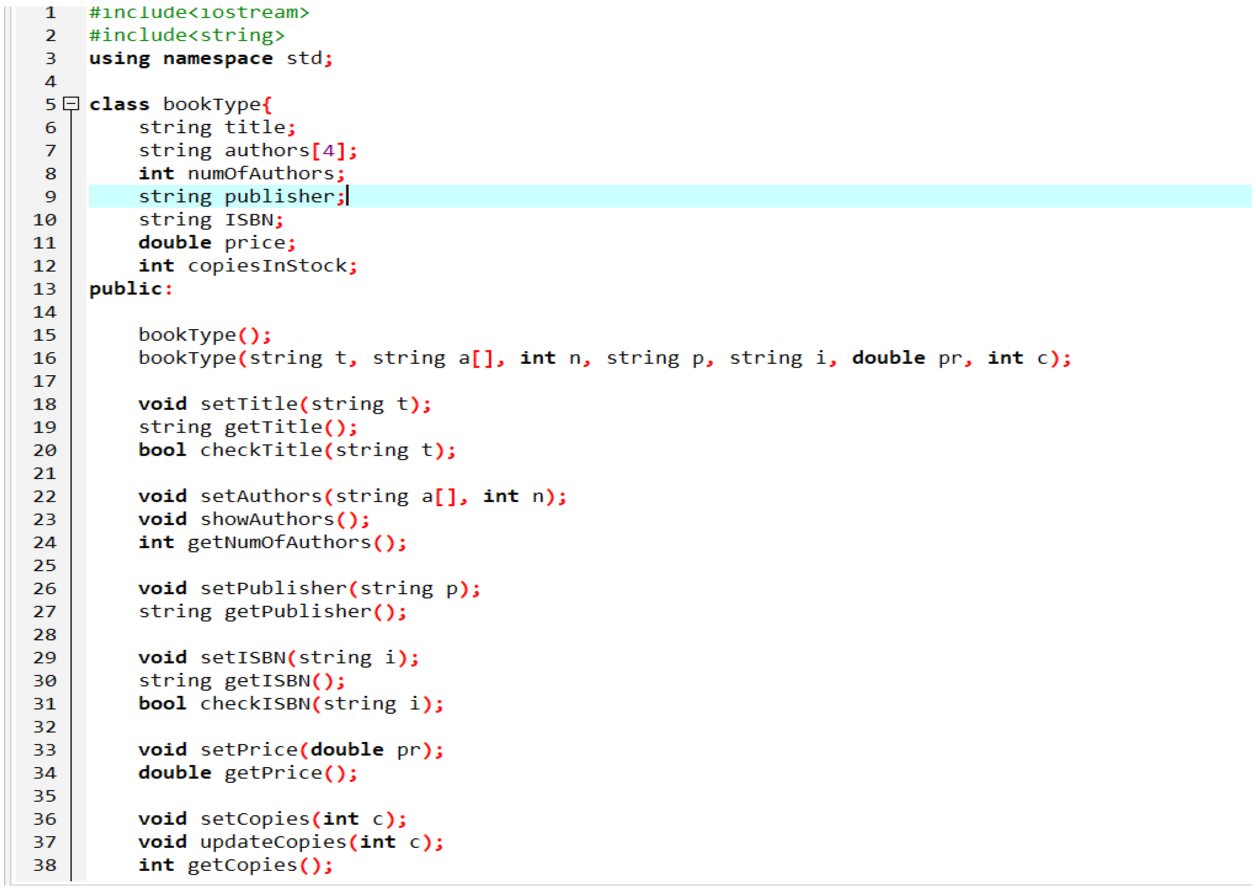
3. Some of the characteristics of a book are the title, author(s), publisher, ISBN, price, and year of publication. Design a class bookType that defines the book as an ADT.

 Each object of the class bookType can hold the following information about a book: title, up to four authors, publisher, ISBN, price, and number of copies in stock. To keep track of the number of authors, add another member variable.

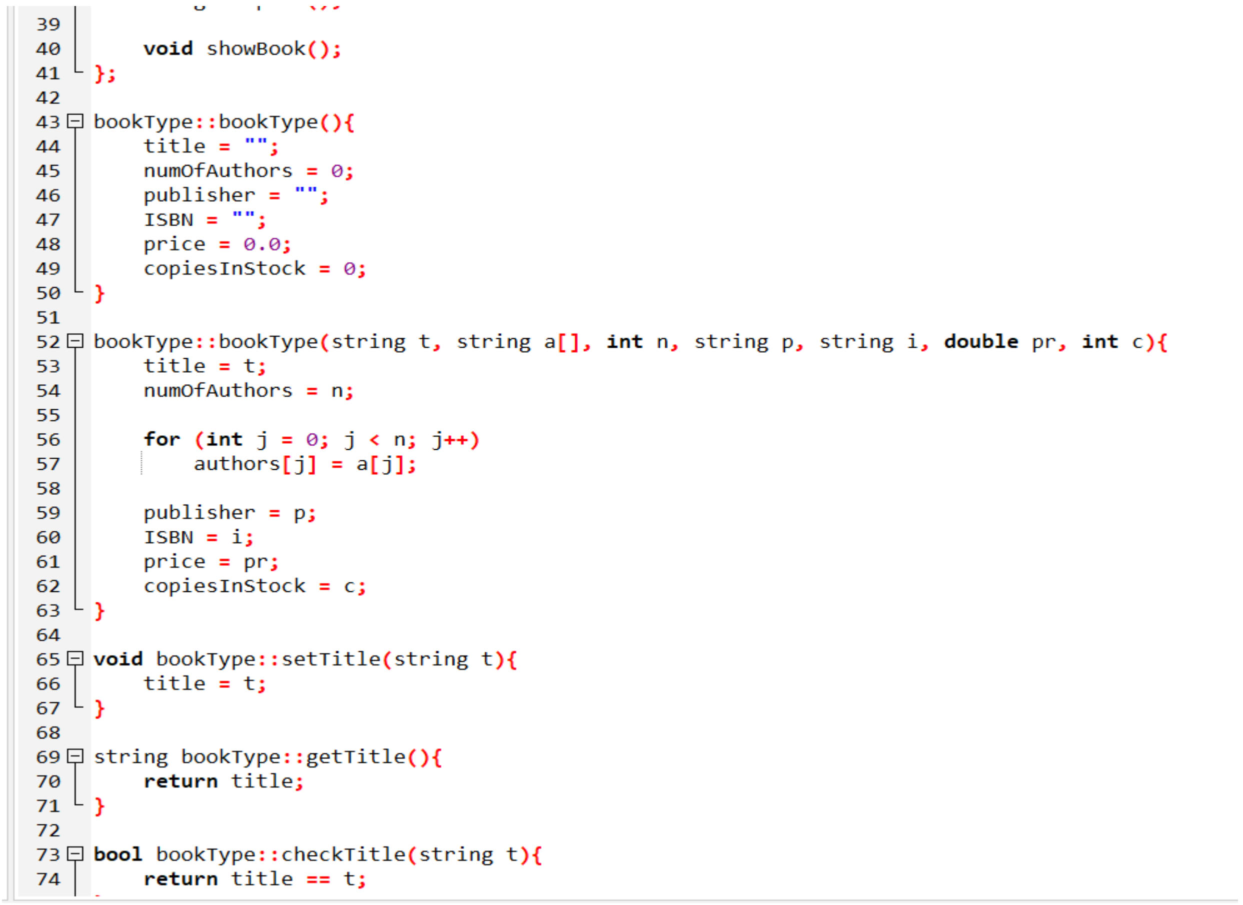
 Include the member functions to perform the various operations on objects of type bookType. For example, the usual operations that can be performed on the title are to show the title, set the title, and check whether a title is the same as the actual title of the book. Similarly, the typical operations that can be performed on the number of copies in stock are to show the number of copies in stock, set the number of copies in stock, update the number of copies in stock, and return the number of copies in stock. Add similar operations for the publisher, ISBN, book price, and authors. Add the appropriate constructors and a destructor (if one is needed).

 Write the definitions of the member functions of the class bookType.

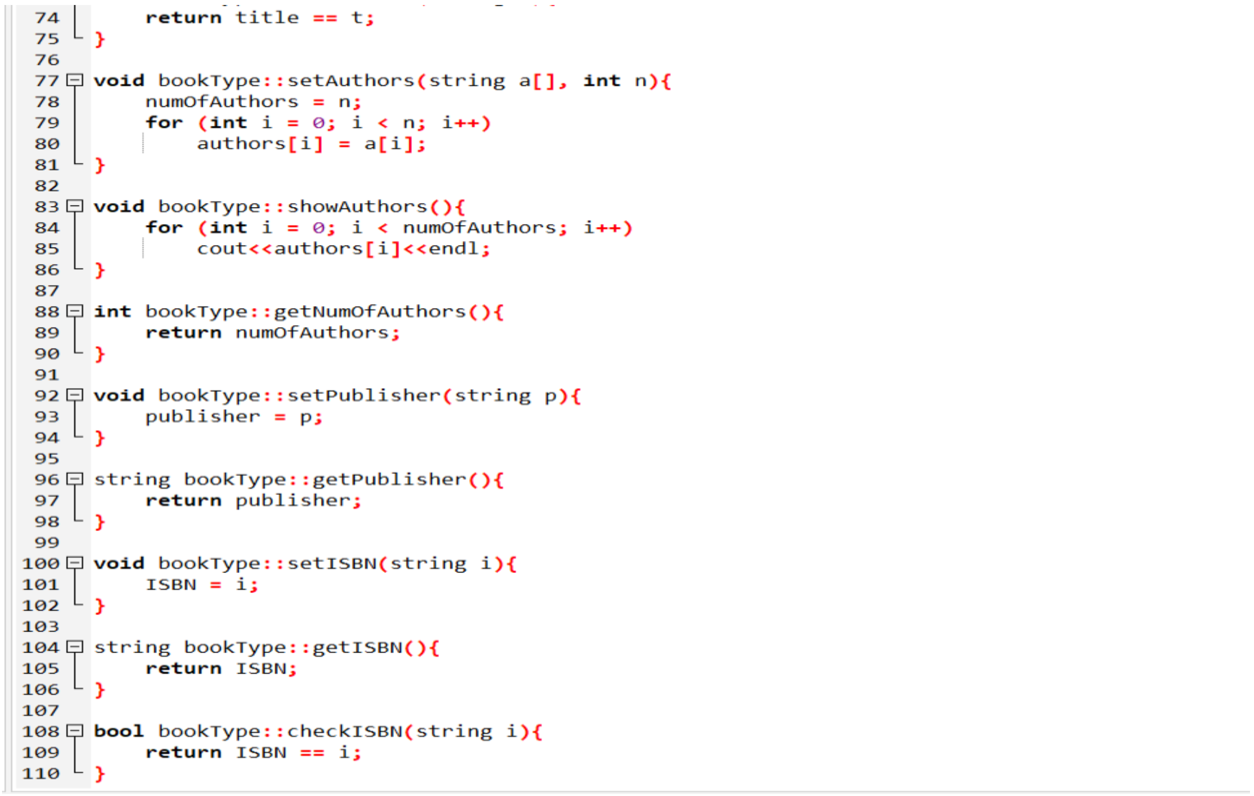
 Write a program that uses the class bookType and tests various operations on the objects of the class bookType. Declare an array of 100 components of type bookType. Some of the operations that you should perform are to search for a book by its title, search by ISBN, and update the number of copies of a book.



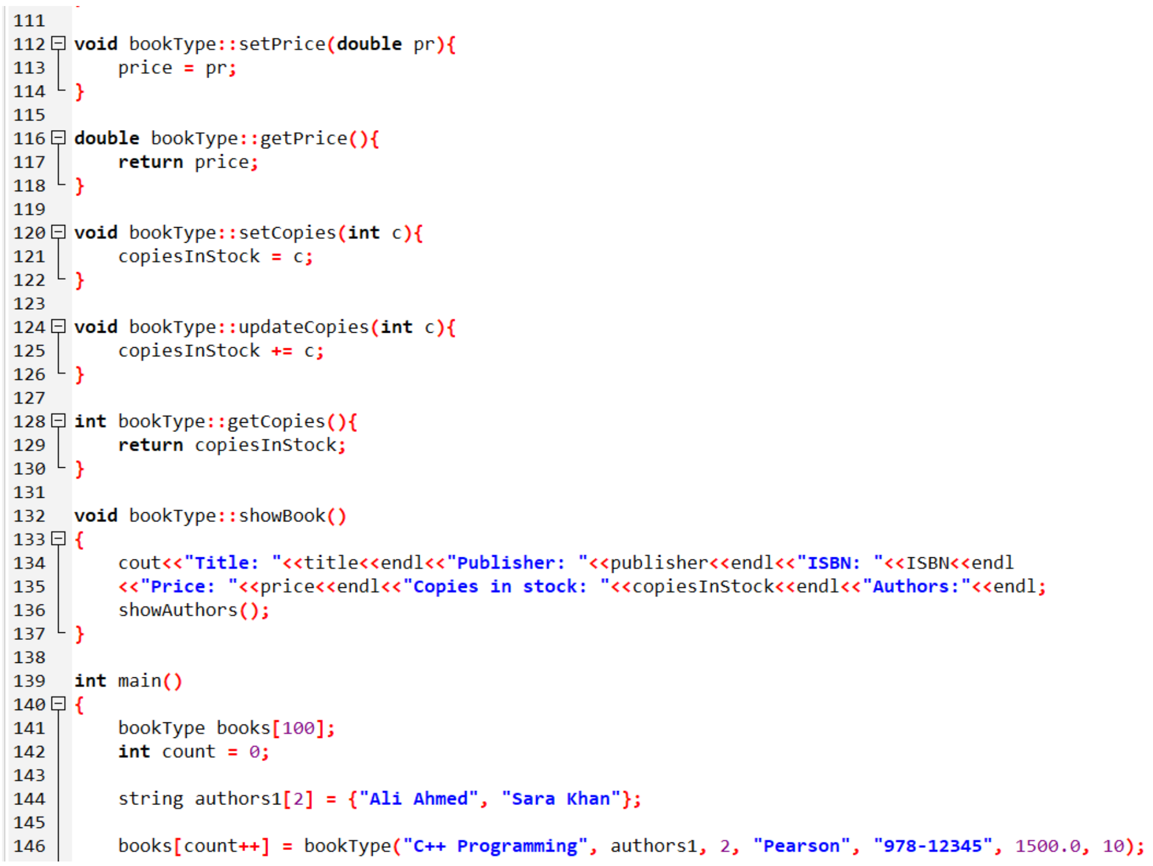
```
#include<iostream>
#include<string>
using namespace std;

class bookType{
    string title;
    string authors[4];
    int numOfAuthors;
    string publisher;
    string ISBN;
    double price;
    int copiesInStock;
public:

    bookType();
    bookType(string t, string a[], int n, string p, string i, double pr, int c);

    void setTitle(string t);
    string getTitle();
    bool checkTitle(string t);

    void setAuthors(string a[], int n);
    void showAuthors();
    int getNumOfAuthors();

    void setPublisher(string p);
    string getPublisher();

    void setISBN(string i);
    string getISBN();
    bool checkISBN(string i);

    void setPrice(double pr);
    double getPrice();

    void setCopies(int c);
    void updateCopies(int c);
    int getCopies();
```



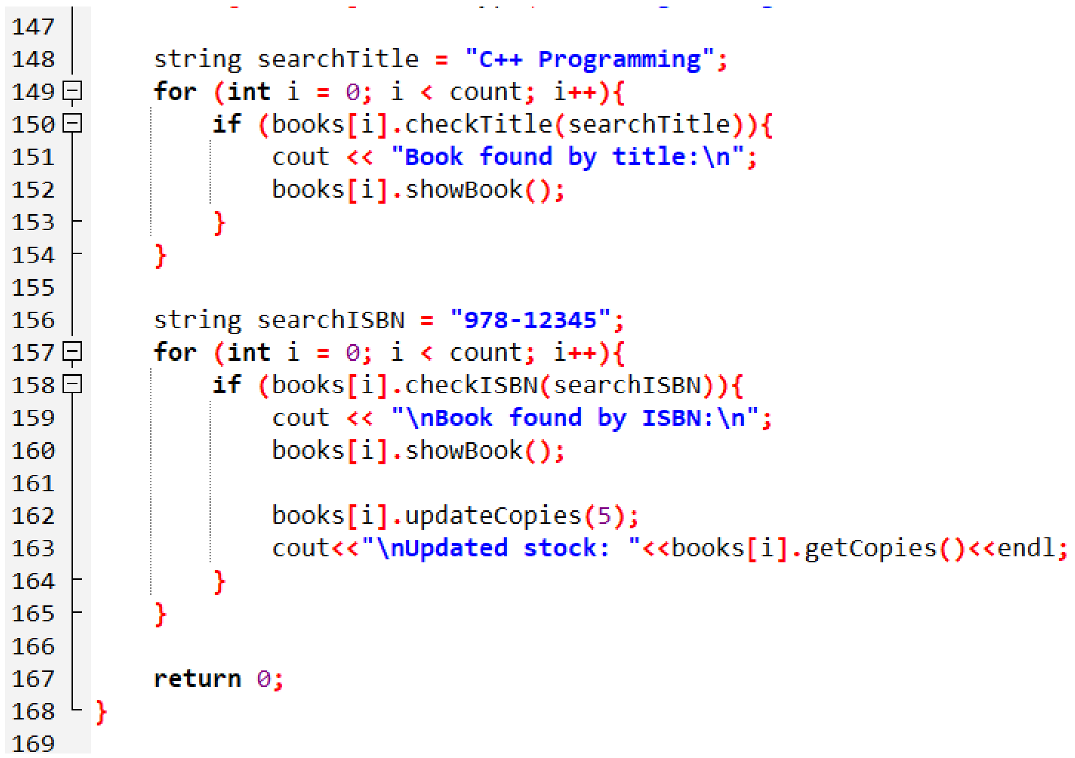
```cpp

    void showBook();
};

bookType::bookType(){
    title = "";
    numOfAuthors = 0;
    publisher = "";
    ISBN = "";
    price = 0.0;
    copiesInStock = 0;
}

bookType::bookType(string t, string a[], int n, string p, string i, double pr, int c){
    title = t;
    numOfAuthors = n;

    for (int j = 0; j < n; j++)
        authors[j] = a[j];

    publisher = p;
    ISBN = i;
    price = pr;
    copiesInStock = c;
}

void bookType::setTitle(string t){
    title = t;
}

string bookType::getTitle(){
    return title;
}

bool bookType::checkTitle(string t){
    return title == t;
```



```cpp
    return title == t;
}

void bookType::setAuthors(string a[], int n){
    numOfAuthors = n;
    for (int i = 0; i < n; i++)
        authors[i] = a[i];
}

void bookType::showAuthors(){
    for (int i = 0; i < numOfAuthors; i++)
        cout<<authors[i]<<endl;
}

int bookType::getNumOfAuthors(){
    return numOfAuthors;
}

void bookType::setPublisher(string p){
    publisher = p;
}

string bookType::getPublisher(){
    return publisher;
}

void bookType::setISBN(string i){
    ISBN = i;
}

string bookType::getISBN(){
    return ISBN;
}

bool bookType::checkISBN(string i){
    return ISBN == i;
}
```

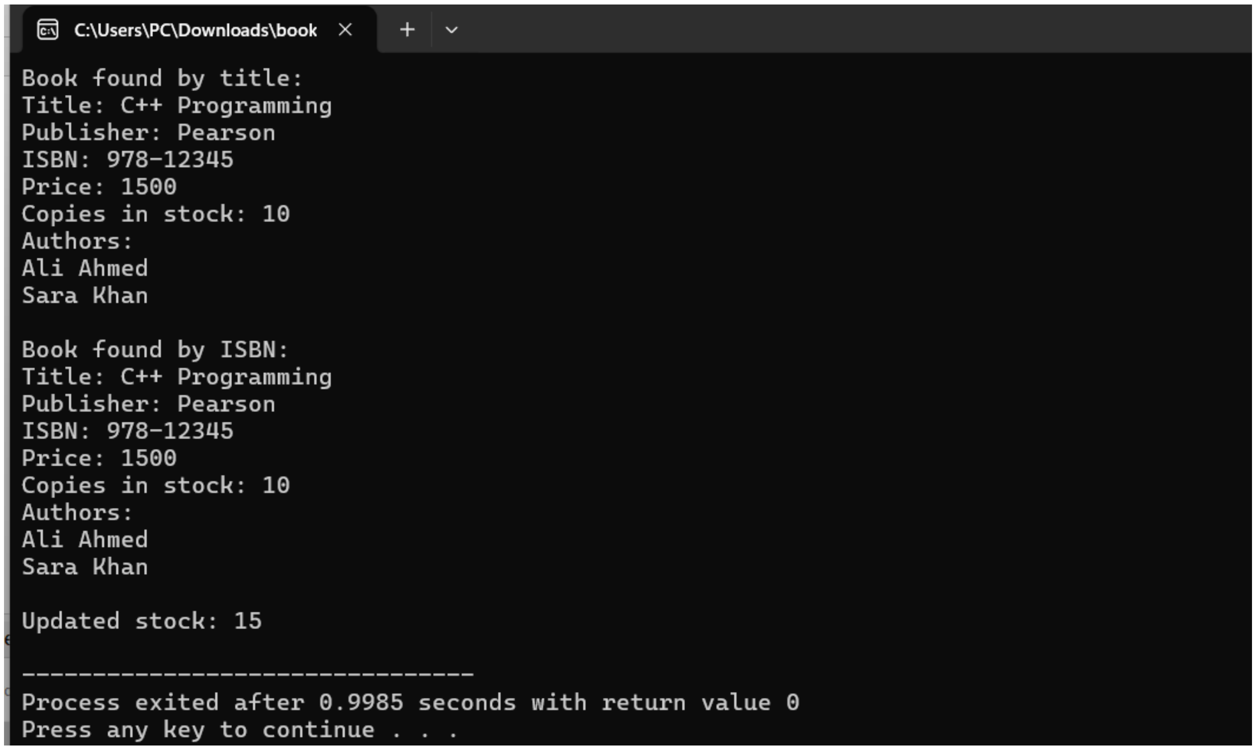


```cpp

void bookType::setPrice(double pr){
    price = pr;
}

double bookType::getPrice(){
    return price;
}

void bookType::setCopies(int c){
    copiesInStock = c;
}

void bookType::updateCopies(int c){
    copiesInStock += c;
}

int bookType::getCopies(){
    return copiesInStock;
}

void bookType::showBook()
{
    cout<<"Title: "<<title<<endl<<"Publisher: "<<publisher<<endl<<"ISBN: "<<ISBN<<endl
    <<"Price: "<<price<<endl<<"Copies in stock: "<<copiesInStock<<endl<<"Authors:"<<endl;
    showAuthors();
}

int main()
{
    bookType books[100];
    int count = 0;

    string authors1[2] = {"Ali Ahmed", "Sara Khan"};

    books[count++] = bookType("C++ Programming", authors1, 2, "Pearson", "978-12345", 1500.0, 10);
```



```cpp

    string searchTitle = "C++ Programming";
    for (int i = 0; i < count; i++){
        if (books[i].checkTitle(searchTitle)){
            cout << "Book found by title:\n";
            books[i].showBook();
        }
    }

    string searchISBN = "978-12345";
    for (int i = 0; i < count; i++){
        if (books[i].checkISBN(searchISBN)){
            cout << "\nBook found by ISBN:\n";
            books[i].showBook();

            books[i].updateCopies(5);
            cout<<"\nUpdated stock: "<<books[i].getCopies()<<endl;
        }
    }

    return 0;
}

```

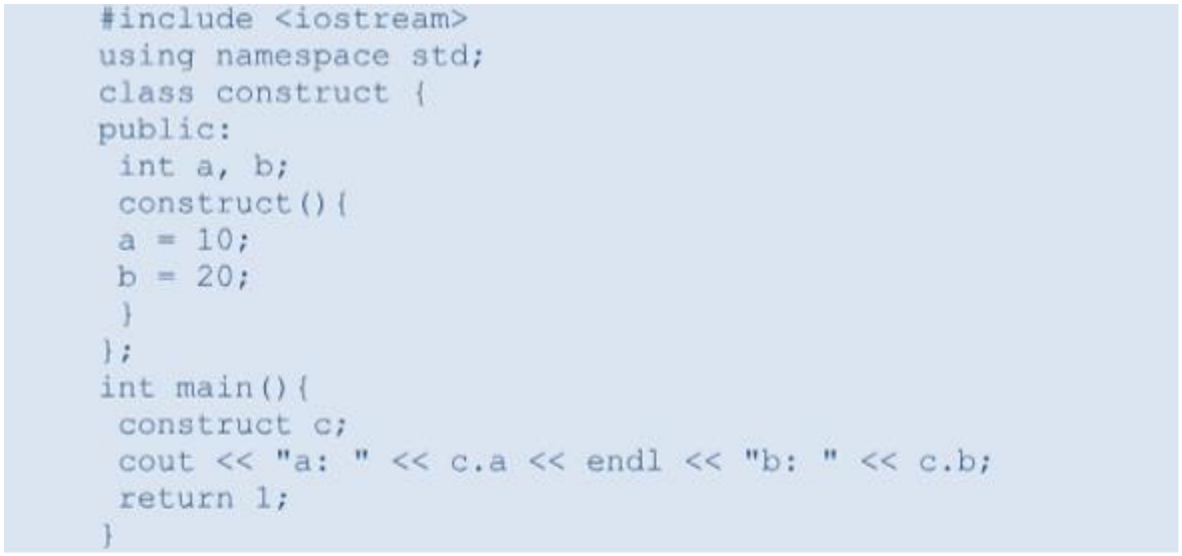
Areeba **CT-25070** **Section B**



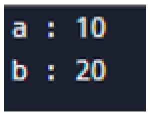
```
Book found by title:
Title: C++ Programming
Publisher: Pearson
ISBN: 978-12345
Price: 1500
Copies in stock: 10
Authors:
Ali Ahmed
Sara Khan

Book found by ISBN:
Title: C++ Programming
Publisher: Pearson
ISBN: 978-12345
Price: 1500
Copies in stock: 10
Authors:
Ali Ahmed
Sara Khan

Updated stock: 15

--------------------------------
Process exited after 0.9985 seconds with return value 0
Press any key to continue . . .
```

<u>EXAMPLES</u>

<u>Example 1:</u>



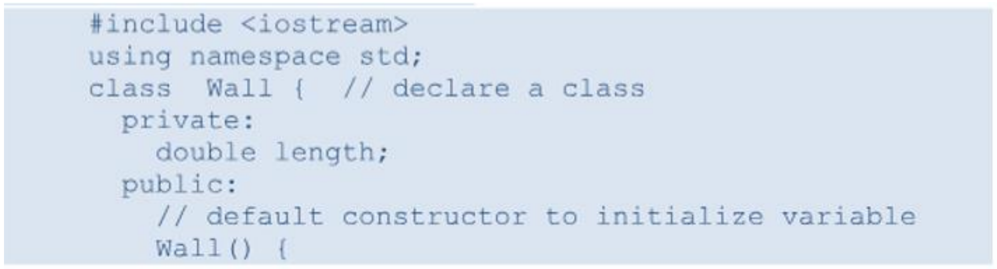
```
#include <iostream>
using namespace std;
class construct {
public:
 int a, b;
 construct(){
 a = 10;
 b = 20;
 }
};
int main(){
 construct c;
 cout << "a: " << c.a << endl << "b: " << c.b;
 return 1;
}
```



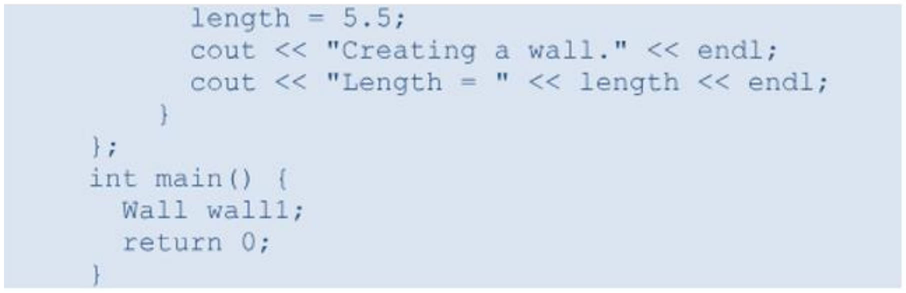
```
a : 10
b : 20
```

Areeba **CT-25070** **Section B**

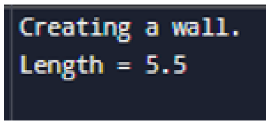
Example 2:



```
#include <iostream>
using namespace std;
class  Wall {  // declare a class
  private:
    double length;
  public:
    // default constructor to initialize variable
    Wall() {
```

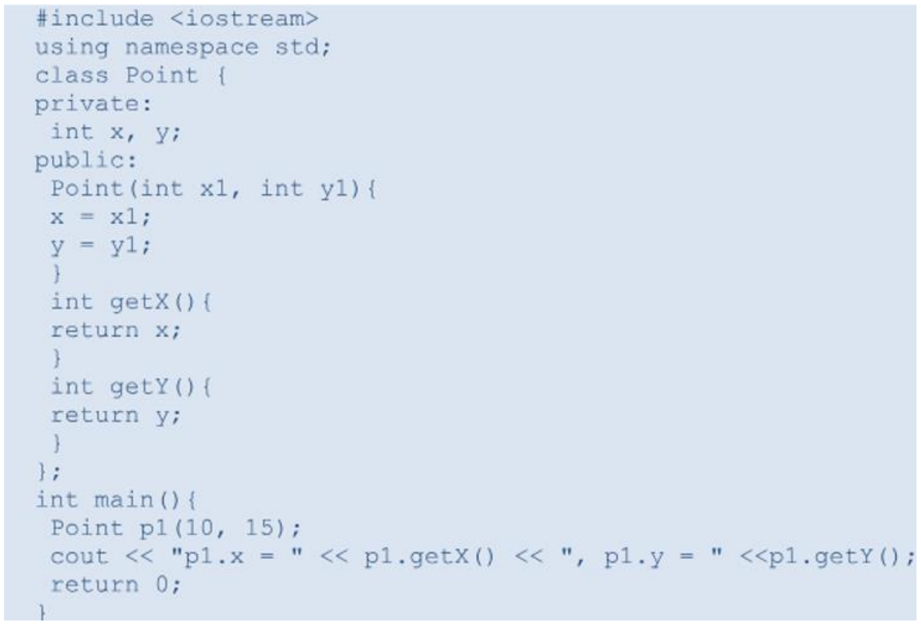


```
      length = 5.5;
      cout << "Creating a wall." << endl;
      cout << "Length = " << length << endl;
    }
};
int main() {
  Wall wall1;
  return 0;
}
```

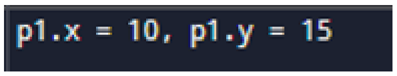


```
Creating a wall.
Length = 5.5
```

Example 3:

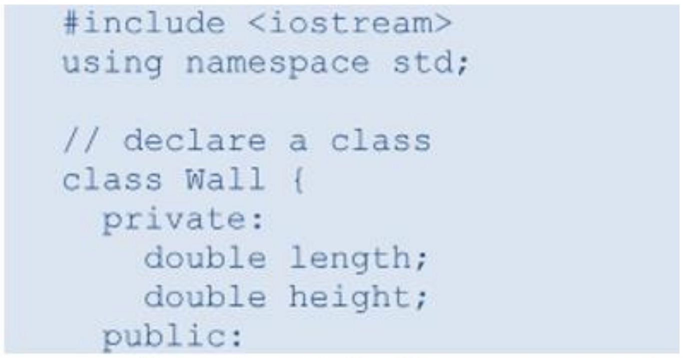
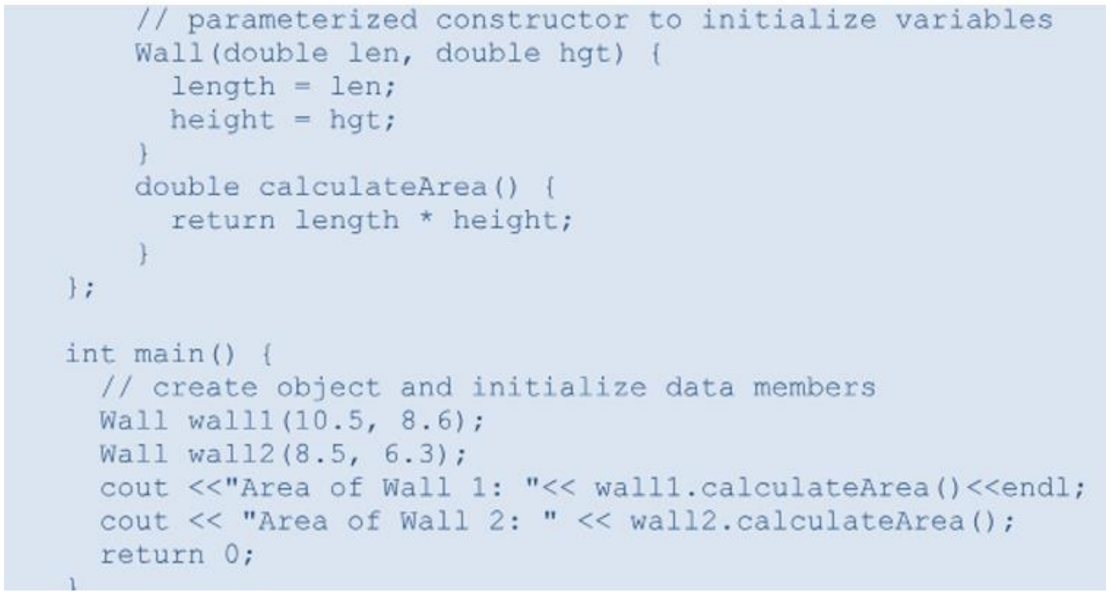


```
#include <iostream>
using namespace std;
class Point {
private:
 int x, y;
public:
 Point(int x1, int y1){
 x = x1;
 y = y1;
 }
 int getX(){
 return x;
 }
 int getY(){
 return y;
 }
};
int main(){
 Point p1(10, 15);
 cout << "p1.x = " << p1.getX() << ", p1.y = " <<p1.getY();
 return 0;
}
```

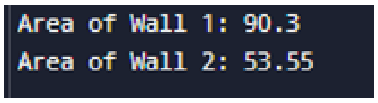


```
p1.x = 10, p1.y = 15
```

<u>**Example 4:**</u>

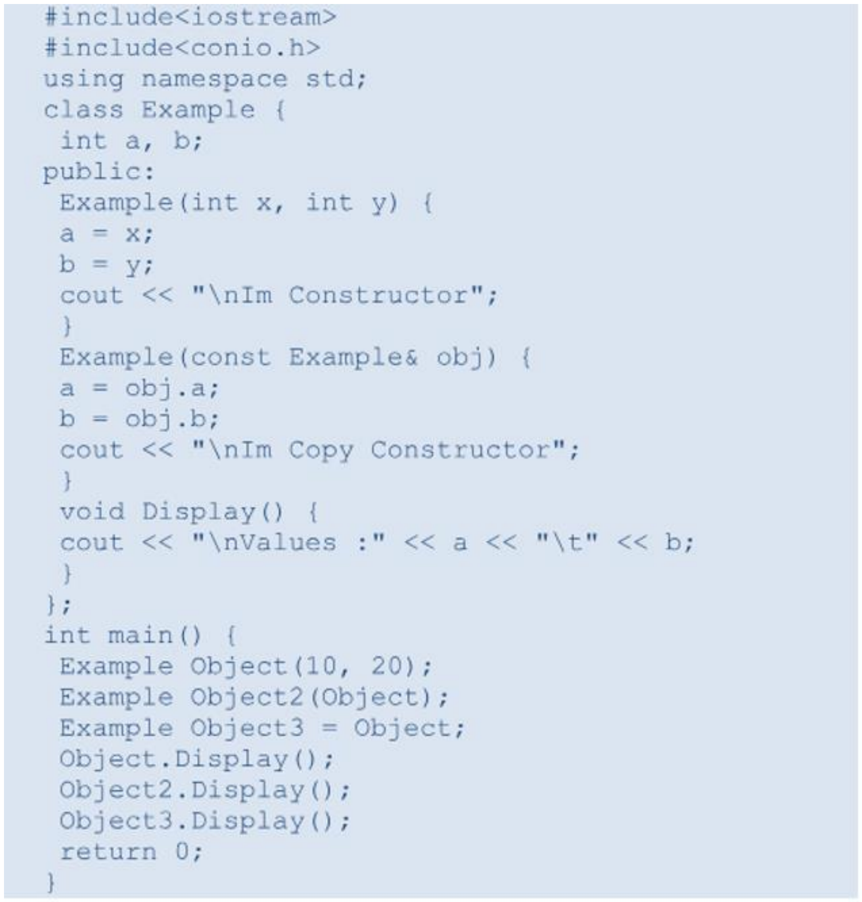
```
#include <iostream>
using namespace std;

// declare a class
class Wall {
  private:
    double length;
    double height;
  public:
    // parameterized constructor to initialize variables
    Wall(double len, double hgt) {
      length = len;
      height = hgt;
    }
    double calculateArea() {
      return length * height;
    }
};

int main() {
  // create object and initialize data members
  Wall wall1(10.5, 8.6);
  Wall wall2(8.5, 6.3);
  cout <<"Area of Wall 1: "<< wall1.calculateArea()<<endl;
  cout << "Area of Wall 2: " << wall2.calculateArea();
  return 0;
}
```



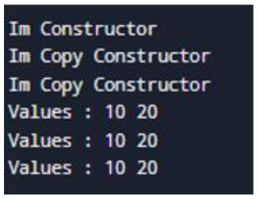
```
Area of Wall 1: 90.3
Area of Wall 2: 53.55
```

Areeba **CT-25070** **Section B**

Example 5:



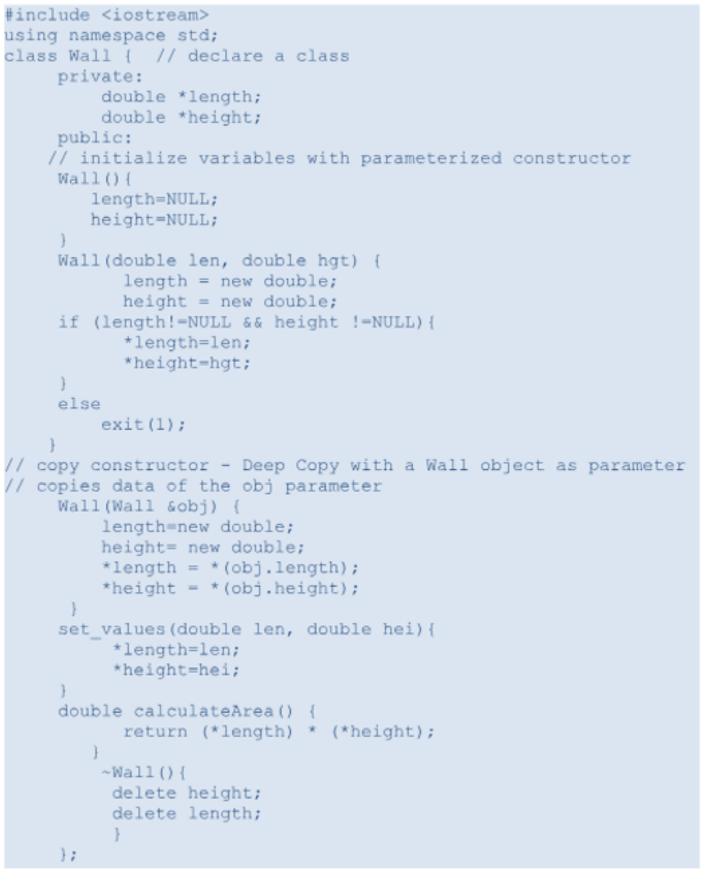
```
#include<iostream>
#include<conio.h>
using namespace std;
class Example {
 int a, b;
public:
 Example(int x, int y) {
 a = x;
 b = y;
 cout << "\nIm Constructor";
 }
 Example(const Example& obj) {
 a = obj.a;
 b = obj.b;
 cout << "\nIm Copy Constructor";
 }
 void Display() {
 cout << "\nValues :" << a << "\t" << b;
 }
};
int main() {
 Example Object(10, 20);
 Example Object2(Object);
 Example Object3 = Object;
 Object.Display();
 Object2.Display();
 Object3.Display();
 return 0;
}
```



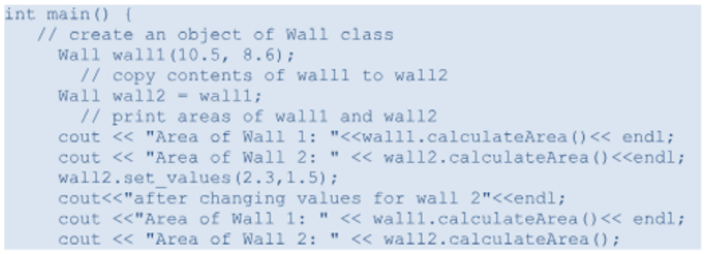
```
Im Constructor
Im Copy Constructor
Im Copy Constructor
Values : 10 20
Values : 10 20
Values : 10 20
```

Areeba **CT-25070** **Section B**

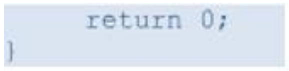
Example 6:



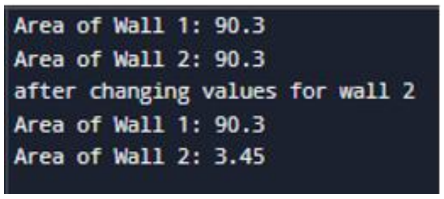
```
#include <iostream>
using namespace std;
class Wall {  // declare a class
     private:
         double *length;
         double *height;
     public:
    // initialize variables with parameterized constructor
     Wall(){
        length=NULL;
        height=NULL;
     }
     Wall(double len, double hgt) {
           length = new double;
           height = new double;
     if (length!=NULL && height !=NULL){
           *length=len;
           *height=hgt;
     }
     else
         exit(1);
    }
// copy constructor - Deep Copy with a Wall object as parameter
// copies data of the obj parameter
     Wall(Wall &obj) {
         length=new double;
         height= new double;
         *length = *(obj.length);
         *height = *(obj.height);
      }
     set_values(double len, double hei){
          *length=len;
          *height=hei;
     }
     double calculateArea() {
           return (*length) * (*height);
        }
         ~Wall(){
          delete height;
          delete length;
           }
     };
```



```
int main() {
   // create an object of Wall class
     Wall wall1(10.5, 8.6);
       // copy contents of wall1 to wall2
     Wall wall2 = wall1;
       // print areas of wall1 and wall2
     cout << "Area of Wall 1: "<<wall1.calculateArea()<< endl;
     cout << "Area of Wall 2: " << wall2.calculateArea()<<endl;
     wall2.set_values(2.3,1.5);
     cout<<"after changing values for wall 2"<<endl;
     cout <<"Area of Wall 1: " << wall1.calculateArea()<< endl;
     cout << "Area of Wall 2: " << wall2.calculateArea();
```

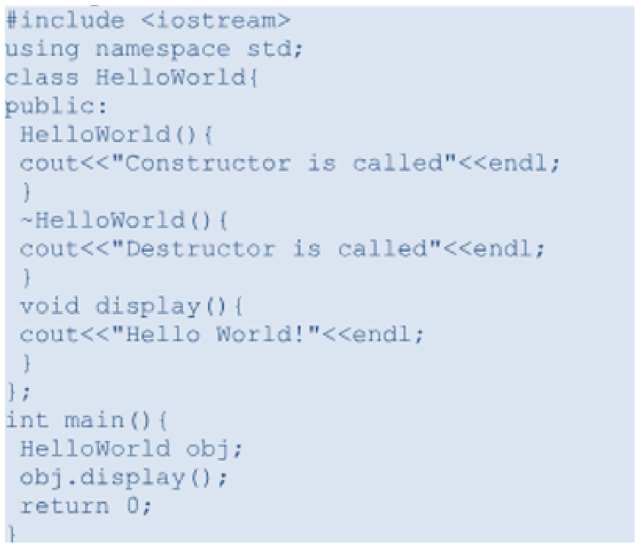


```
     return 0;
}
```

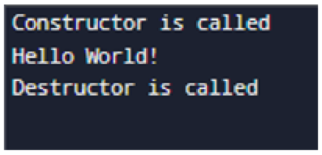


```
Area of Wall 1: 90.3
Area of Wall 2: 90.3
after changing values for wall 2
Area of Wall 1: 90.3
Area of Wall 2: 3.45
```

Example 7:



```
#include <iostream>
using namespace std;
class HelloWorld{
public:
 HelloWorld(){
 cout<<"Constructor is called"<<endl;
 }
 ~HelloWorld(){
 cout<<"Destructor is called"<<endl;
 }
 void display(){
 cout<<"Hello World!"<<endl;
 }
};
int main(){
 HelloWorld obj;
 obj.display();
 return 0;
}
```



```
Constructor is called
Hello World!
Destructor is called
```