OBJECT ORIENTED PROGRAMING

OOP

PRACTICALS

NAME: SYEDA FAIZA ASLAM

SECTION: B

ROLL NO: CT-25064

LAB NO: 03

COURSE CODE: CT-260

COURSE TEACHER: Mr. AHMED ZAKI

Exercise

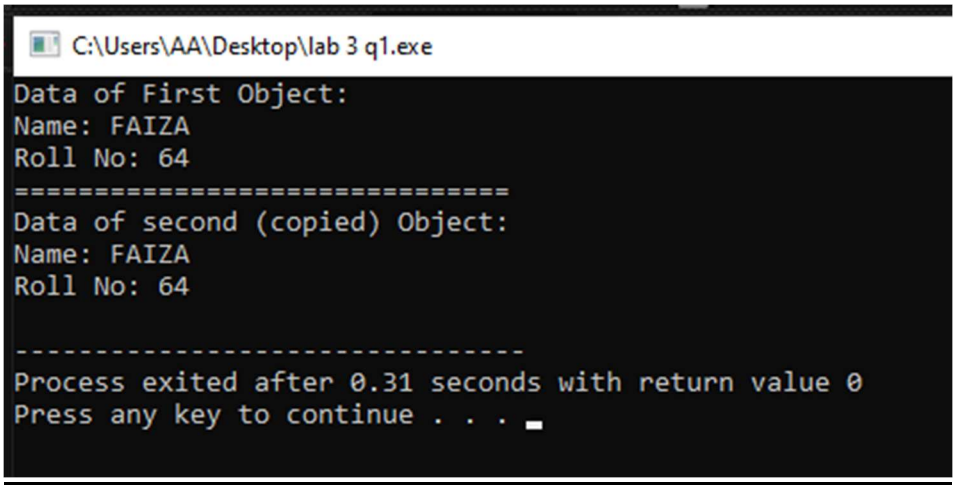
1. Write a C++ program to copy the value of one object to another object using copy constructor.

SOURCE CODE:

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

class Student
{
	private:
		string name;
		int roll_no;
	public:
		Student(string n, int r)
		{
			name = n;
			roll_no = r;
		}
		Student(Student &obj)
		{
			name = obj.name;
			roll_no = obj.roll_no;
		}
		void displayData()
		{
			cout<<"Name: "<<name<<endl;
			cout<<"Roll No: "<<roll_no<<endl;
		}
};
int main()
```

```
{
	Student s1("FAIZA",64);
	cout<<"Data of First Object: \n";
	s1.displayData();
	cout<<"==============================\n";

	Student s2(s1);
	cout<<"Data of second (copied) Object: \n";
	s2.displayData();

	return 0;

}
```

OUTPUT:



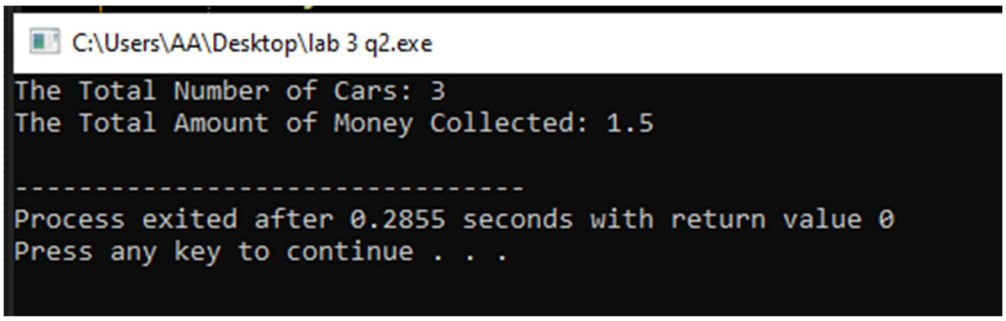
2.Create a class tollbooth. The two data items are a type int to hold the total number of cars and a type double to hold the total amount of money collected. A constructor initializes both these to 0. When a car passes the toll, a member function called payingCar( ) increments the car total and adds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

SOURCE CODE:

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

class Tollbooth
{
	private:
		int tC;
		double tM;
	public:
		Tollbooth()
		{
			tC = 0;
			tM = 0;
		}
		void payingCar()
		{
			tC++;
			tM+=0.50;
		}
		void displayData()
		{
			cout<<"The Total Number of Cars: "<<tC<<endl;
```

```
                cout<<"The Total Amount of Money Collected: "<<tM<<endl;
            }
};
int main()
{
    Tollbooth t1;
    t1.payingCar();
    t1.payingCar();
    t1.payingCar();

    t1.displayData();

    return 0;
}
```

OUTPUT:



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.

SOURCE CODE:

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

class bookType
{
```

```
private:
   string title;
   string author[4];
   int noOfAuthors;
   string publisher;
   string ISBN;
   double price;
   int copies;

public:
   // Default Constructor
   bookType()
   {
      title = “”;
      publisher = “”;
      ISBN = “”;
      price = 0;
      copies = 0;
      noOfAuthors = 0;
   }

   // Parameterized Constructor
   bookType(string t, string a[], int n, string pub, string isbn, double p, int c)
   {
      title = t;
      publisher = pub;
      ISBN = isbn;
      price = p;
      copies = c;

      if (n > 4)
         n = 4;

      noOfAuthors = n;
```

```
   for (int I = 0; I < noOfAuthors; i++)
   {
      author[i] = a[i];
   }
}

// Set Functions
void setTitle(string t)
{
   title = t;
}

void setPublisher(string pub)
{
   publisher = pub;
}

void setISBN(string isbn)
{
   ISBN = isbn;
}

void setPrice(double p)
{
   price = p;
}

void setCopies(int c)
{
   copies = c;
}

void setAuthors(string a[], int n)
{
   if (n > 4)
```

```
      n = 4;

   noOfAuthors = n;

   for (int I = 0; I < noOfAuthors; i++)
   {
      author[i] = a[i];
   }
}

// Get Functions
string getTitle()
{
   return title;
}

string getPublisher()
{
   return publisher;
}

string getISBN()
{
   return ISBN;
}

double getPrice()
{
   return price;
}

int getCopies()
{
   return copies;
}
```

```
int getNoOfAuthors()
{
   return noOfAuthors;
}

// Check Functions
bool checkTitle(string t)
{
   if (title == t)
      return true;
   else
      return false;
}

bool checkISBN(string isbn)
{
   if (ISBN == isbn)
      return true;
   else
      return false;
}

// Update Copies
void updateCopies(int c)
{
   copies = copies + c;
}

// Show Authors
void showAuthors()
{
   cout << “Authors: “;
   for (int I = 0; I < noOfAuthors; i++)
   {
```

```
            cout << author[i];
            if (I != noOfAuthors – 1)
                cout << “, “;
        }
        cout << endl;
    }

    // Print Book Details
    void print()
    {
        cout << “\n--------------------------\n”;
        cout << “Title: “ << title << endl;
        showAuthors();
        cout << “Publisher: “ << publisher << endl;
        cout << “ISBN: “ << ISBN << endl;
        cout << “Price: “ << price << endl;
        cout << “Copies in Stock: “ << copies << endl;
        cout << “--------------------------\n”;
    }
};

// Search Book by Title
int searchTitle(bookType b[], int size, string t)
{
    for (int I = 0; I < size; i++)
    {
        if (b[i].checkTitle(t) == true)
            return I;
    }
    return -1;
}

// Search Book by ISBN
int searchISBN(bookType b[], int size, string isbn)
{
```

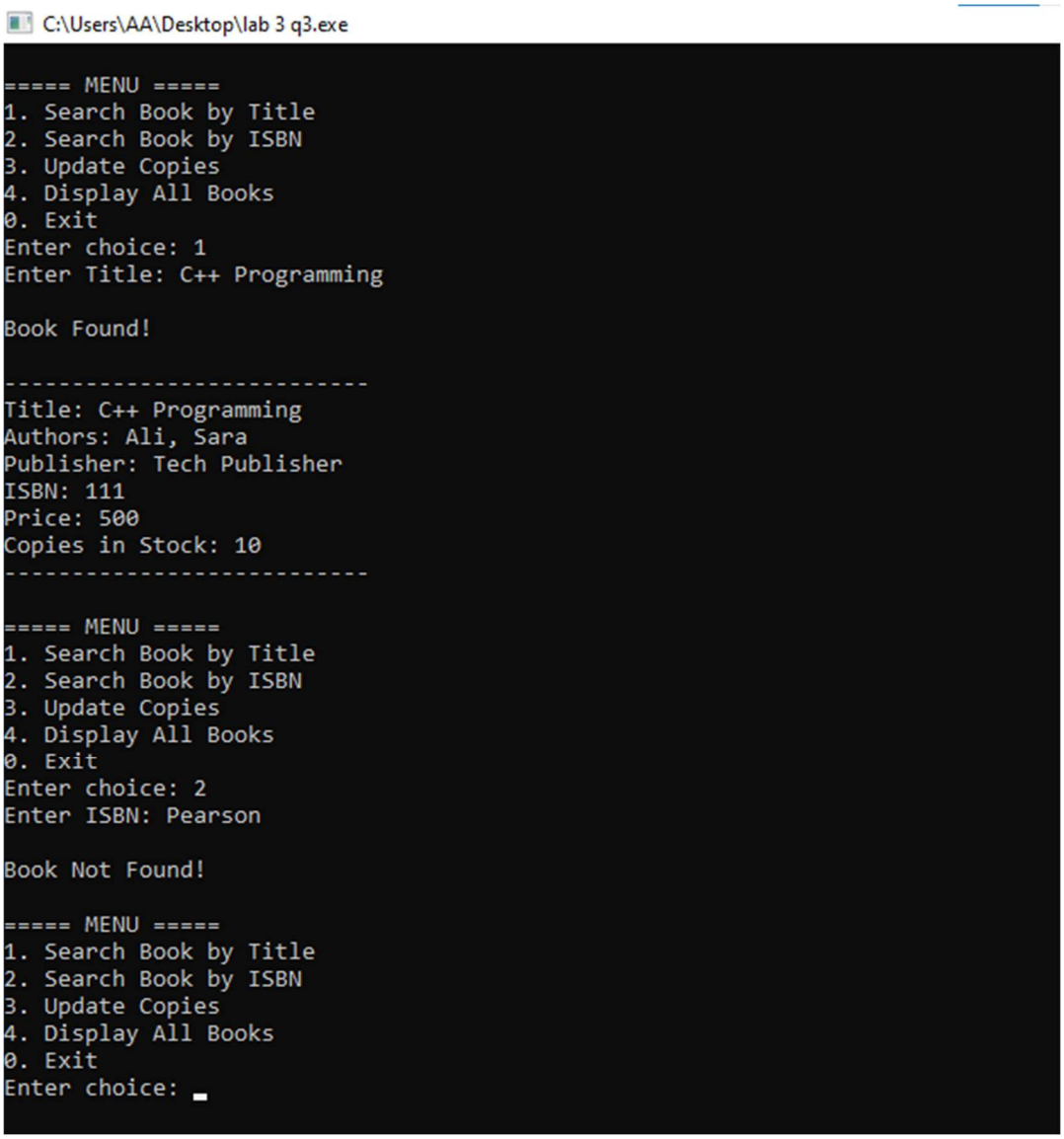
```
    for (int I = 0; I < size; i++)
    {
        if (b[i].checkISBN(isbn) == true)
            return I;
    }
    return -1;
}

int main()
{
    bookType book[100];
    int totalBooks = 3;

    // Sample Books
    string a1[2] = {“Ali”, “Sara”};
    book[0] = bookType(“C++ Programming”, a1, 2, “Tech Publisher”, “111”, 500,
10);

    string a2[1] = {“John”};
    book[1] = bookType(“Data Structures”, a2, 1, “Oxford”, “222”, 700, 5);

    string a3[3] = {“David”, “Emma”, “Noah”};
    book[2] = bookType(“Artificial Intelligence”, a3, 3, “Pearson”, “333”, 1200, 7);

    int choice;
    string t, isbn;
    int addCopies;

    do
    {
        cout << “\n===== MENU =====\n”;
        cout << “1. Search Book by Title\n”;
        cout << “2. Search Book by ISBN\n”;
        cout << “3. Update Copies\n”;
        cout << “4. Display All Books\n”;
```

```
cout << “0. Exit\n”;
cout << “Enter choice: “;
cin >> choice;
cin.ignore();

if (choice == 1)
{
   cout << “Enter Title: “;
   getline(cin, t);

   int index = searchTitle(book, totalBooks, t);

   if (index != -1)
   {
      cout << “\nBook Found!\n”;
      book[index].print();
   }
   else
   {
      cout << “\nBook Not Found!\n”;
   }
}

else if (choice == 2)
{
   cout << “Enter ISBN: “;
   getline(cin, isbn);

   int index = searchISBN(book, totalBooks, isbn);

   if (index != -1)
   {
      cout << “\nBook Found!\n”;
      book[index].print();
   }
```

```
    else
    {
        cout << "\nBook Not Found!\n";
    }
}

else if (choice == 3)
{
    cout << "Enter ISBN of Book: ";
    getline(cin, isbn);

    int index = searchISBN(book, totalBooks, isbn);

    if (index != -1)
    {
        cout << "Enter copies to add: ";
        cin >> addCopies;

        book[index].updateCopies(addCopies);

        cout << "\nCopies Updated Successfully!\n";
        book[index].print();
    }
    else
    {
        cout << "\nBook Not Found!\n";
    }
}

else if (choice == 4)
{
    for (int I = 0; I < totalBooks; i++)
    {
        book[i].print();
    }
```

```
        }

    } while (choice != 0);

    cout << "\nProgram Ended.\n";
    return 0;
}
```

OUTPUT:



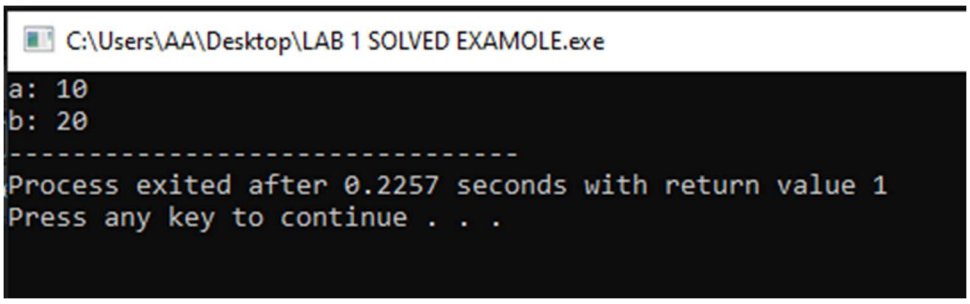
EXAMPLES:

Example: 1

SOURCE CODE:

```
#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;
}
```

OUTPUT:

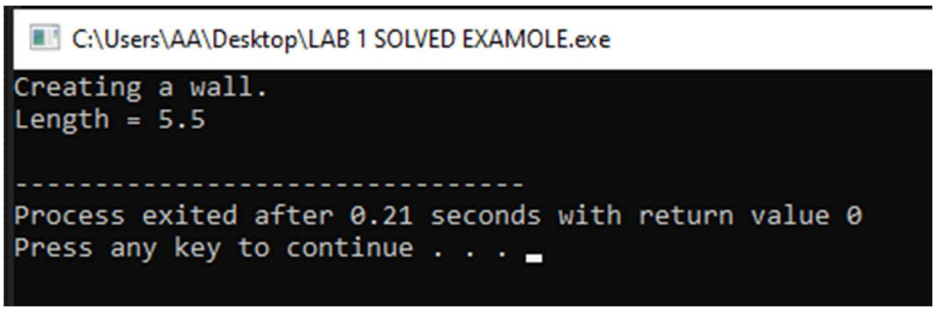


Example: 2

SOURCE CODE:

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

class Wall
{
	private:
		double length;
	public:
		Wall()
		{
			length = 5.5;
			cout<<"Creating a wall."<<endl;
			cout<<"Length = "<<length<<endl;
		}
};

int main()
{
	Wall wall1;
	return 0;
}
```

OUTPUT:

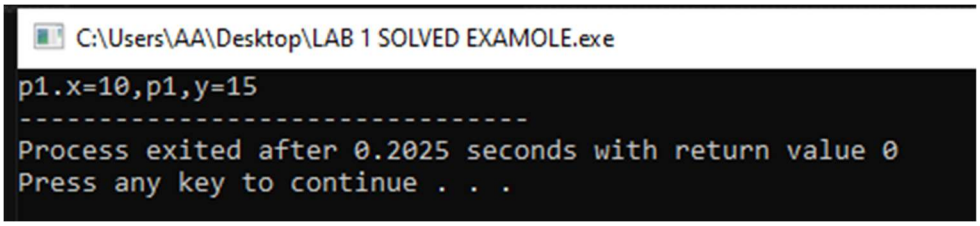


Example: 3

SOURCE CODE:

```
#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;
}
```

OUTPUT:



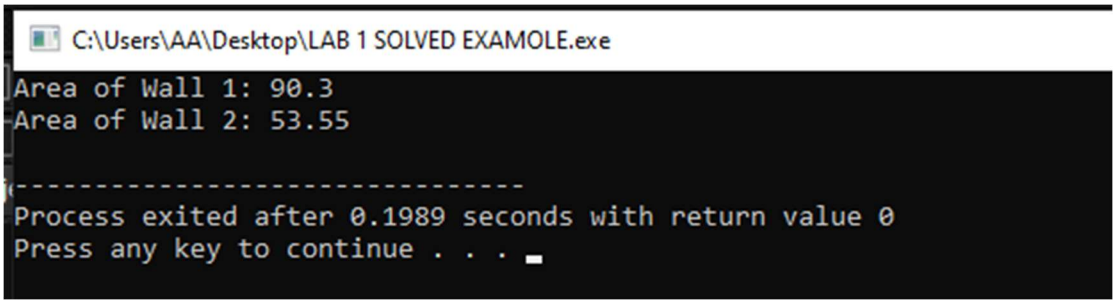
Example: 4

SOURCE CODE:

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

// Declare a class
class Wall {
private:
   double length;
   double height;

public:
   // Parameterized constructor
   Wall(double len, double hgt) {
      length = len;
      height = hgt;
   }

   // Member function to calculate area
   double calculateArea() {
      return length * height;
   }
};
```

```
int main() {
   // Create objects
   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() << endl;

   return 0;
}
```

OUTPUT:



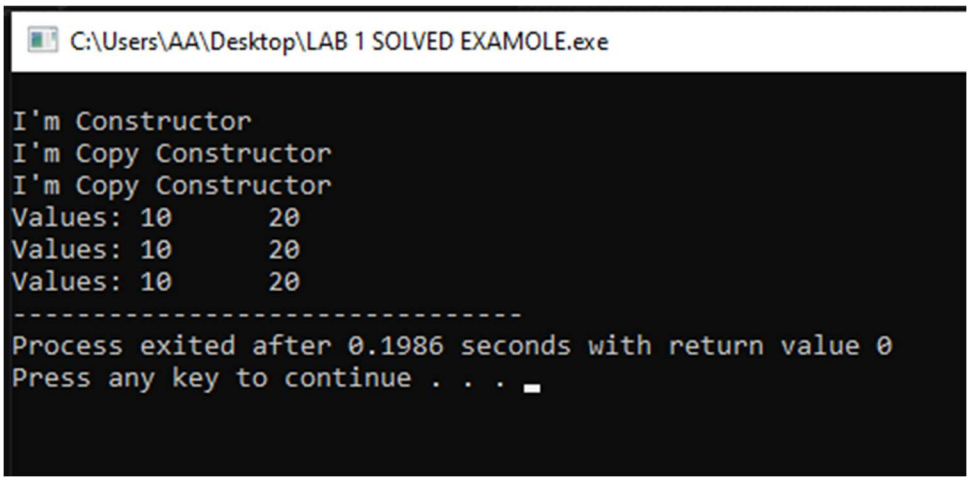
Example: 5

SOURCE CODE:

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

class Example {
   int a, b;
public:
   Example(int x, int y) {      // Parameterized constructor
      a = x;
      b = y;
      cout << "\nI'm Constructor";
```

```
    }

    Example(const Example &obj) { // Copy constructor
        a = obj.a;
        b = obj.b;
        cout << "\nI'm Copy Constructor";
    }

    void Display() {
        cout << "\nValues: " << a << "\t" << b;
    }
};

int main() {
    Example Object(10, 20);
    Example Object2(Object);  // Calls copy constructor
    Example Object3 = Object; // Also calls copy constructor

    Object.Display();
    Object2.Display();
    Object3.Display();
    return 0;
}
```

OUTPUT:



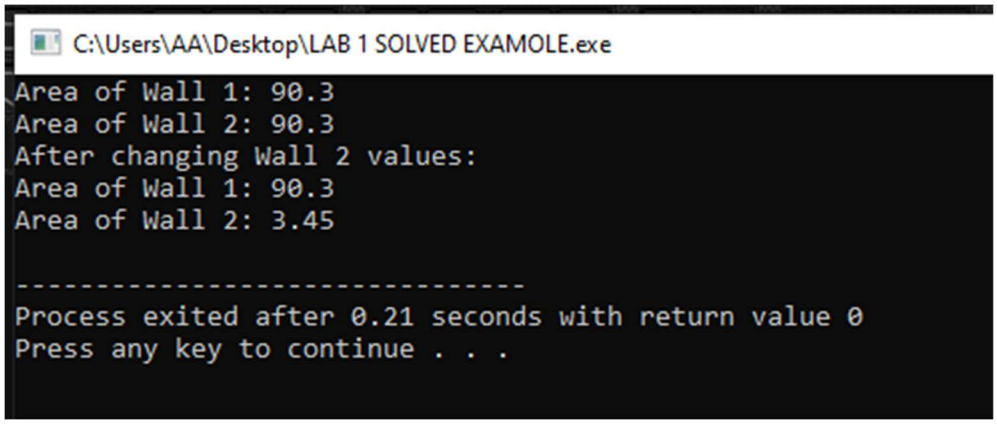
Example: 6

SOURCE CODE:

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

class Wall {
private:
   double *length;
   double *height;
public:
   Wall() { length = height = NULL; }

   Wall(double len, double hgt) {   // Parameterized constructor
      length = new double(len);
      height = new double(hgt);
   }

   Wall(const Wall &obj) {         // Copy constructor (deep copy)
      length = new double(*(obj.length));
      height = new double(*(obj.height));
   }

   void set_values(double len, double hei) {
      *length = len;
      *height = hei;
   }

   double calculateArea() {
      return (*length) * (*height);
   }

   ~Wall() {                    // Destructor
```

```
        delete length;
        delete height;
    }
};

int main() {
    Wall wall1(10.5, 8.6);
    Wall wall2 = wall1; // deep copy

    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 Wall 2 values:" << endl;
    cout << "Area of Wall 1: " << wall1.calculateArea() << endl;
    cout << "Area of Wall 2: " << wall2.calculateArea() << endl;

    return 0;
}
```

OUTPUT:



Example: 7

SOURCE CODE:

```
#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;
}
```

OUTPUT: