Subject:- C++
LAB Assignment - 5

1. Write a program for the default constructor.

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

// Class definition
class Rectangle {
private:
    int length;
    int width;

public:
    // Default constructor
    Rectangle() {
        length = 0;
        width = 0;
    }
```

Subject:- C++

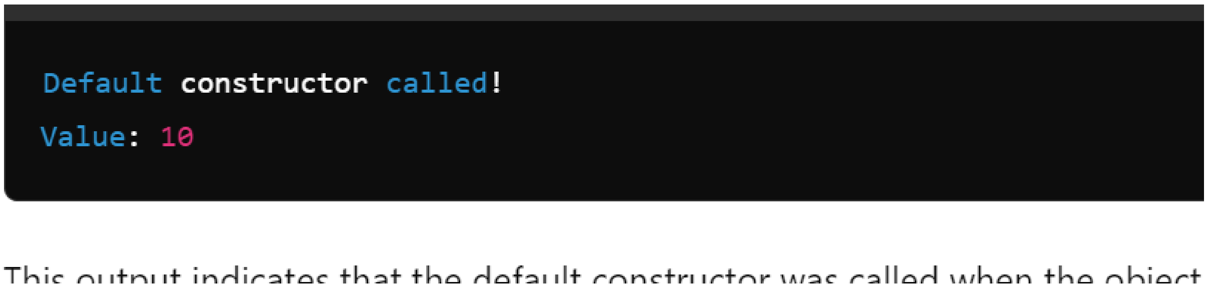
LAB Assignment - 5

1. Write a program for the default constructor.

#include <iostream>

```
using namespace std;


// Class definition

class Rectangle {

private:

    int length;

    int width;


public:

    // Default constructor

    Rectangle() {

        length = 0;

        width = 0;
```

Output:



```
Default constructor called!
Value: 10
```

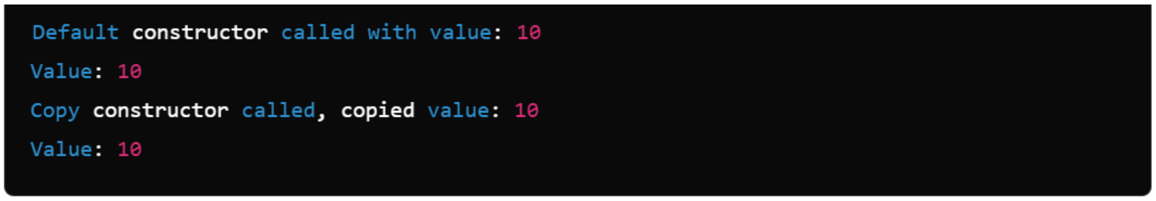
This output indicates that the default constructor was called when the object

2) Write a program for copy constructors.

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

// Class definition
```

```
class Rectangle {
private:
    int length;
    int width;

public:
    // Parameterized constructor
    Rectangle(int l, int w) {
        length = l;
        width = w;
    }

    // Copy constructor
    Rectangle(const Rectangle &rect) {
        length = rect.length;
        width = rect.width;
    }

    // Member function to display the values
    void display() {
        cout << "Length: " << length << ", Width: " << width << endl;
    }
};

int main() {
    // Creating an object using parameterized constructor
    Rectangle rect1(10, 5);

    // Creating a new object using the copy constructor
    Rectangle rect2 = rect1;  // Copying rect1 into rect2

    // Display the values of both objects
    cout << "Rectangle 1: ";
    rect1.display();
    cout << "Rectangle 2: ";
    rect2.display();

    return 0;
}
```

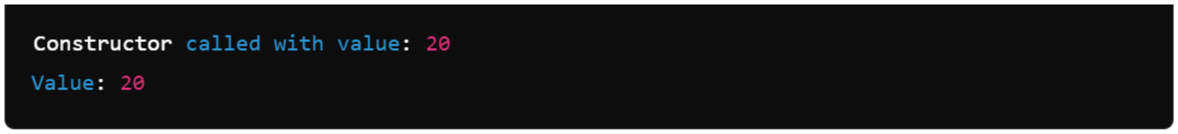
Output:



```
Default constructor called with value: 10
Value: 10
Copy constructor called, copied value: 10
Value: 10
```

3) Write a program for Dynamic Initialization of objects.

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

class Rectangle {
private:
    int length;
    int width;

public:
    // Constructor for dynamic initialization
    Rectangle(int l, int w) {
        length = l;
        width = w;
    }
    // Member function to calculate the area
    int area() {
        return length * width;
    }

    // Member function to display the dimensions
    void display() {
        cout << "Length: " << length << ", Width: " << width << endl;
    }
};

int main() {
    int l, w;

    // Taking input from the user
    cout << "Enter the length of the rectangle: ";
    cin >> l;

    cout << "Enter the width of the rectangle: ";
    cin >> w;

    // Dynamically initializing the object with user input
    Rectangle *rect = new Rectangle(l, w);

    // Displaying dimensions and calculating the area
    rect->display();
    cout << "Area of the rectangle: " << rect->area() << endl;

    // Deallocating the dynamically allocated memory
    delete rect;

    return 0;
}
```
Output:

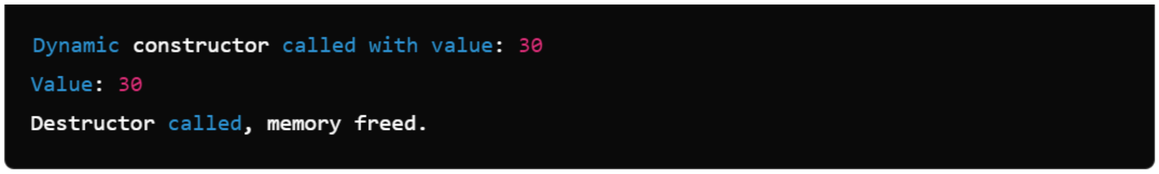


```
Constructor called with value: 20
Value: 20
```

4) Write a program for Dynamic Constructors.

```
#include <iostream>
#include <cstring>  // For strcpy and strlen
using namespace std;

class String {
private:
    char *name;
    int length;

public:
    // Dynamic constructor
    String(const char *str) {
        length = strlen(str);  // Calculate the length of the string
        name = new char[length + 1];  // Dynamically allocate memory for the
string
        strcpy(name, str);  // Copy the string into the allocated memory
    }

    // Destructor to deallocate memory
    ~String() {
        delete[] name;  // Free dynamically allocated memory
        cout << "Memory released." << endl;
    }
    // Member function to display the string
    void display() const {
        cout << "String: " << name << endl;
    }
};

int main() {
    // Creating object using dynamic constructor
    String str1("Hello, World!");

    // Displaying the dynamically created string
    str1.display();

    return 0;
}
```

Output:



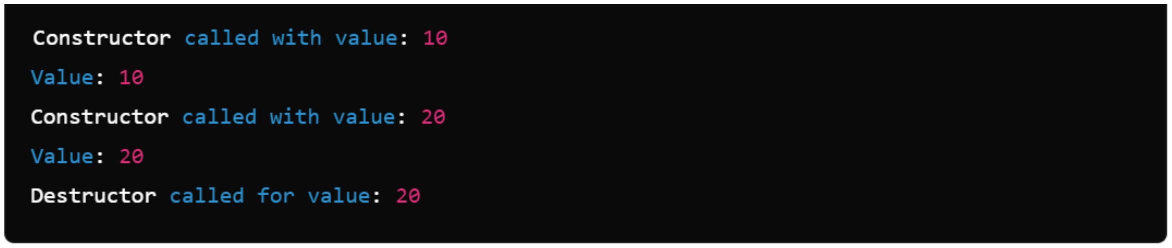
```
Dynamic constructor called with value: 30
Value: 30
Destructor called, memory freed.
```

5) Write a program for the Destructor.

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

class Rectangle {
private:
    int length;
    int width;
```

```
public:
    // Constructor to initialize the object
    Rectangle(int l, int w) {
        length = l;
        width = w;
        cout << "Rectangle created with length = " << length << " and width = "
<< width << endl;
    }

    // Destructor
    ~Rectangle() {
        cout << "Destructor called. Rectangle with length = " << length << "
and width = " << width << " is destroyed." << endl;
    }

    // Member function to calculate the area
    int area() const {
        return length * width;
    }

    // Member function to display the rectangle's dimensions
    void display() const {
        cout << "Length: " << length << ", Width: " << width << endl;
    }
};

int main() {
    // Creating a Rectangle object
    Rectangle rect1(10, 5);

    // Displaying the rectangle's details
    rect1.display();
    cout << "Area of the rectangle: " << rect1.area() << endl;
    // The destructor is automatically called when the object goes out of
scope (at the end of main)
    return 0;
}
```

Output:

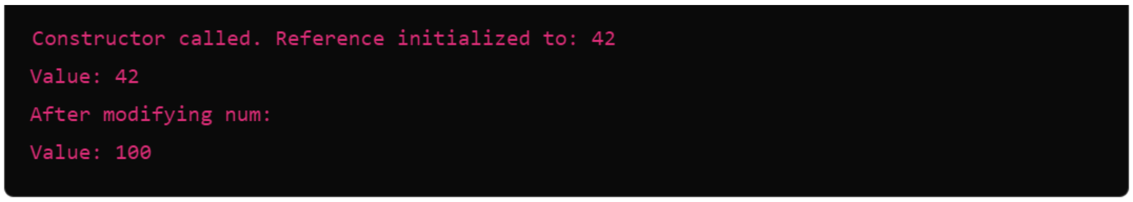


```
Constructor called with value: 10
Value: 10
Constructor called with value: 20
Value: 20
Destructor called for value: 20
```

6) Write a program of initializing reference members in a class using a
     member initializer list.

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

class Rectangle {
private:
    int &length;  // Reference member
    int &width;   // Reference member

public:
    // Constructor with member initializer list to initialize reference members
    Rectangle(int &l, int &w) : length(l), width(w) {
        cout << "Rectangle created with length = " << length << " and width = "
<< width << endl;
    }

    // Function to display the dimensions
    void display() const {
        cout << "Length: " << length << ", Width: " << width << endl;
    }
};

int main() {
    int len = 10;
    int wid = 5;

    // Creating a Rectangle object with reference to existing variables
    Rectangle rect(len, wid);

    // Displaying the rectangle's dimensions
    rect.display();

    // Modifying the original variables
    len = 15;
    wid = 7;

    // Displaying the updated dimensions (since reference members reflect
the changes)
    rect.display();

    return 0;
}
```

Output:

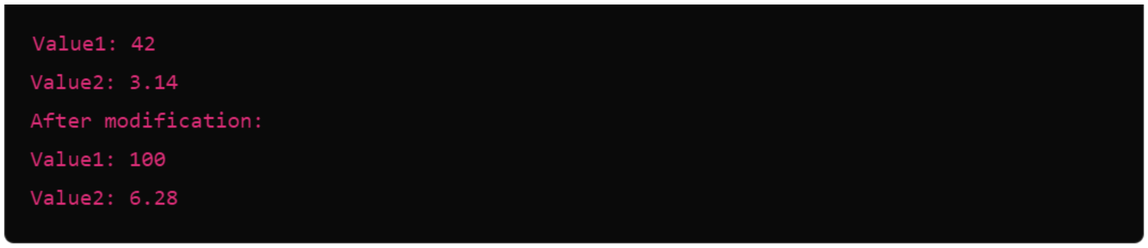


```
Constructor called. Reference initialized to: 42
Value: 42
After modifying num:
Value: 100
```

7) Write a program of pointers to Data Members.

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

class Rectangle {
public:
    int length;  // Public data member
    int width;   // Public data member

    // Constructor to initialize the object
    Rectangle(int l, int w) : length(l), width(w) {}

    // Function to display the dimensions
    void display() const {
        cout << "Length: " << length << ", Width: " << width << endl;
    }
};

int main() {
    // Creating an object of the Rectangle class
    Rectangle rect(10, 5);

    // Declaring pointers to data members of class Rectangle
    int Rectangle::*ptrLength = &Rectangle::length;
    int Rectangle::*ptrWidth = &Rectangle::width;

    // Accessing and displaying the data members using the pointers
    cout << "Using pointer to data members:" << endl;
    cout << "Length: " << rect.*ptrLength << endl;
    cout << "Width: " << rect.*ptrWidth << endl;

    // Modifying the data members using the pointers
    rect.*ptrLength = 15;
    rect.*ptrWidth = 7;

    // Display the updated dimensions
    cout << "\nAfter modifying using pointer to data members:" << endl;
    rect.display();

    return 0;
}
```

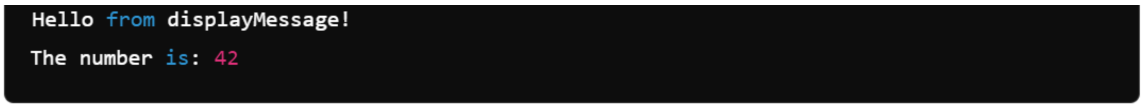
Output:



```
Value1: 42
Value2: 3.14
After modification:
Value1: 100
Value2: 6.28
```

8) Write a program of pointer to Member Function.

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

class Rectangle {
private:
    int length;
    int width;

public:
    // Constructor to initialize the object
    Rectangle(int l, int w) : length(l), width(w) {}
    // Member function to calculate the area
    int area() const {
        return length * width;
    }

    // Member function to display the dimensions
    void display() const {
        cout << "Length: " << length << ", Width: " << width << endl;
    }
};

int main() {
    // Creating an object of the Rectangle class
    Rectangle rect(10, 5);

    // Declaring a pointer to member function for the display function
    void (Rectangle::*ptrDisplay)() const = &Rectangle::display;

    // Declaring a pointer to member function for the area function
    int (Rectangle::*ptrArea)() const = &Rectangle::area;

    // Calling member function using pointer to member function
    cout << "Calling display function using pointer:" << endl;
    (rect.*ptrDisplay)();  // Calling display using pointer

    cout << "Calling area function using pointer:" << endl;
    int area = (rect.*ptrArea)();  // Calling area using pointer
    cout << "Area: " << area << endl;

    return 0;
}
```
Output:



```
Hello from displayMessage!
The number is: 42
```