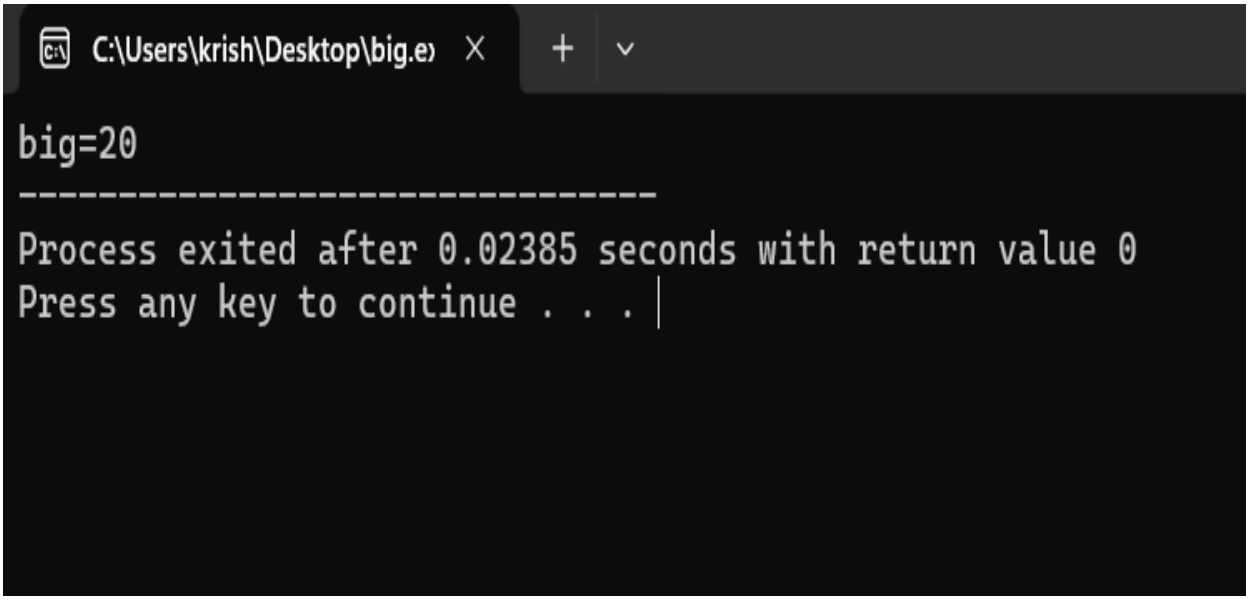
**AIM:** Write a C++ program to illustrate inline functions and function overloading.

**PROGRAM:**

```
#include<iostream>
using namespace std;
// Inline function
inline int big(int a,int b)
{
	return (a>b?a:b);
}
int main()
{
	cout<<"big="<<big(10,20); //printing value by calling inline function
	return 0;
}
```

**OUTPUT:**



**AIM:** Write a C++ program to illustrate inline function and function overloading

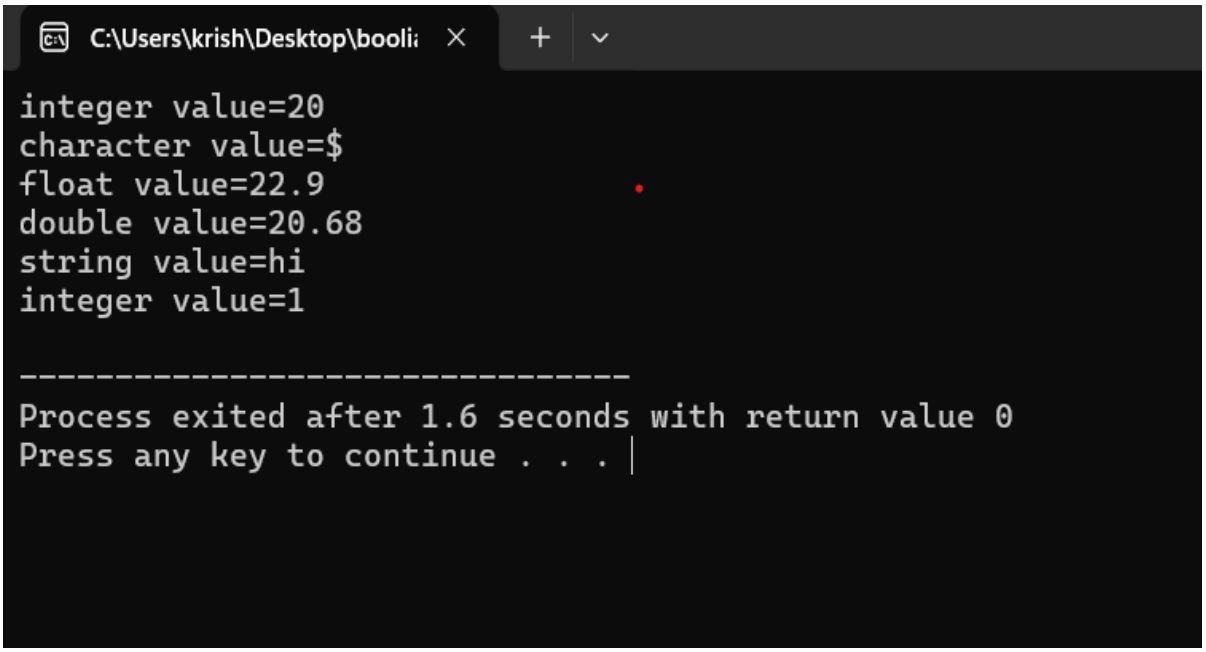
**PROGRAM:**

```
#include<iostream>
using namespace std;
// Define a class named 'sample'
class sample
{
	public:
		 // Overloaded function 'show' to display an integer
		void show(int i)
		{
			cout<<"integer value="<<i<<endl;
		}
		 // Overloaded function 'show' to display a character
		void show(char c)
		{
			cout<<"character value="<<c<<endl;
		}
		  // Overloaded function 'show' to display a float value
		void show(float f)
		{
			cout<<"float value="<<f<<endl;
		}
		  // Overloaded function 'show' to display a string (char pointer)
		void show(char *s)
		{
			cout<<"string value="<<s<<endl;
		}
		 // Overloaded function 'show' to display a double value
```

```
        void show(double d)
        {
                cout<<"double value="<<d<<endl;
        }
         // Overloaded function 'show' to display a boolean value
        void show(bool b)
        {
                cout<<"boolean value="<<b<<endl;
        }

};
int main()
{
        sample s;
        s.show(20);  // Call 'show' with an integer argument → calls show(int)
        s.show('$');  // Call 'show' with a character argument → calls show(char)
        s.show(22.9f);  // Call 'show' with a float argument (22.9f) → calls show(float)
        s.show(20.68);  // Call 'show' with a double argument→ calls show(double)
        s.show("hi");  // Call 'show' with a string literal  → calls show(char*)
        s.show(true);  // Call 'show' with a boolean argument → calls show(bool)
        return 0;
}
```

**OUTPUT:**


C:\Users\krish\Desktop\booli
integer value=20
character value=$
float value=22.9
double value=20.68
string value=hi
integer value=1
--------------------------------
Process exited after 1.6 seconds with return value 0
Press any key to continue . . .

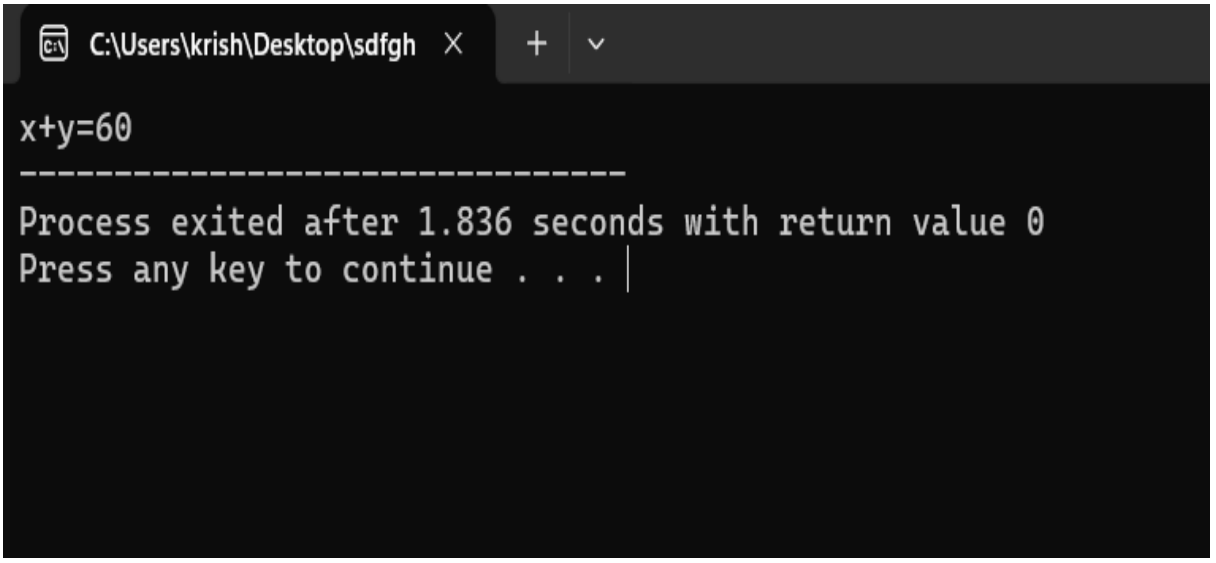
**AIM:** Write a C++ program to Program to illustrate friend function

**PROGRAM:**

```
#include<iostream>
using namespace std;
// Define a class 'Demo
class Demo
{
        private:
                int x;  // Private data member
                  // Private member function to initialize 'x'
                void get()
                {
                        x=55;
                }
                 // Declare 'sum()' as a friend function
                friend void sum();
};
// Friend function definition (outside the class)
void sum()
{
        int y=5;   // Local variable
        Demo d;  // Create object of class Demo
        d.get();  // Call private member function 'get()'
        cout<<"x+y="<<d.x+y;
}
int main()
{
        sum();  // Call the friend function
        return 0;
```

}

**OUTPUT:**


C:\Users\krish\Desktop\sdfgh
x+y=60
--------------------------------
Process exited after 1.836 seconds with return value 0
Press any key to continue . . .

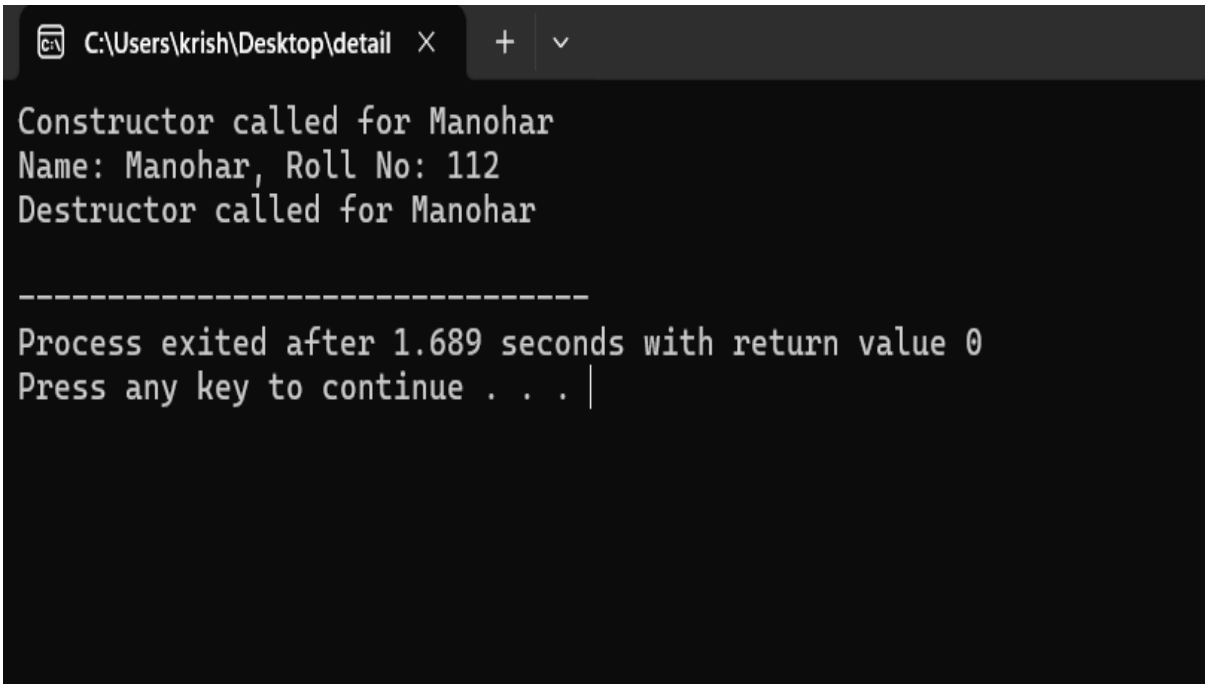
**AIM:** Write a C++ program to illustrate the use of Constructors and Destructors.

**PROGRAM:**

```
#include <iostream>
using namespace std;
// Define a class named 'Sample'
class Sample {
private:
    int rollNo;  // Private data member for student's roll number
    string name;  // Private data member for student's name
public:
// Parameterized constructor
    Sample(int r, string n) {
        rollNo = r;  // Initialize roll number
        name = n;  // Initialize name
        cout << "Constructor called for " << name << endl;
    }
// Member function to display object details
    void show() {
        cout << "Name: " << name << ", Roll No: " << rollNo << endl;
    }
// Destructor
    ~Sample() {
        cout << "Destructor called for " << name << endl;
    }
};
int main() {
// Create object 's1'
    Sample s1(112, "Manohar");
// Call member function to display details
```

```
    s1.show();
    return 0;
}
```

**OUTPUT:**



**AIM:** Write a C++ program illustrating Constructor overloading

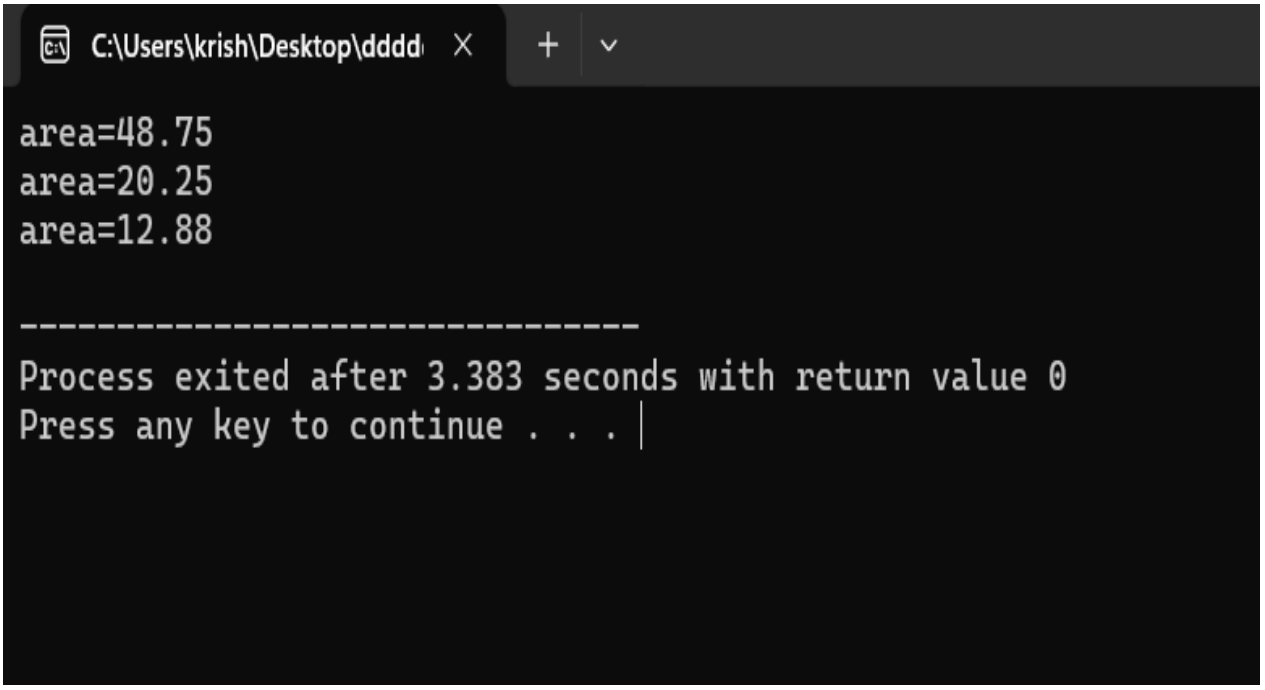
**PROGRAM:**

```
#include<iostream>
using namespace std;
// Define a class 'Rectangle'
class Rectangle
{
        private:
                float length,breadth;  // Private data members
                public:
                        // Default constructor
                Rectangle()
                {           // Initializes length and breadth with fixed values
                        length=7.5;
                        breadth=6.5;
                }
                 // Parameterized constructor with two parameters
                Rectangle(float x,float y)
                {           length=x;
                        breadth=y;
                }
                   // Parameterized constructor with one parameter
                Rectangle(float x)
                {
                        length=x;
                        breadth=x;
                }
                // Member function to calculate and display area
                  void area()
```

```
                {
                        cout<<"area="<<length*breadth<<endl;
                }

};
int main()
{
        Rectangle r1;  // Object created using default constructor
        Rectangle r2(4.5f);  // Object created using one-argument constructor
        Rectangle r3(2.3f,5.6f);  // Object created using two-argument constructor
         //function calls
        r1.area();
        r2.area();
        r3.area();
        return 0;
}
```

**OUTPUT:**



**AIM:** Write a C++ program illustrating Copy Constructor

**PROGRAM:**

```
#include<iostream>
using namespace std;
// Define a class 'sample'
class sample
{
        private:
                int x;  // Private data member
                public:
                        // Default constructor
                        sample()
                        {
                                x=45;
                        }
                        // Copy constructor
                        // Initializes new object with the value of an existing object
                        sample(sample &s1)
                        {
                                x=s1.x;  // Copy the value of 'x' from object s1
                        }
                        // Member function to display the value of 'x'
                        void show()
                        {
                                cout<<"x="<<x<<endl;
                        }
};
int main()
{               // Create object s1 using default constructor
```

```
        sample s1;
        s1.show();
          // Create object s2 using copy constructor
        sample s2(s1);
        s2.show();
        return 0;
}
```

**OUTPUT:**