Lab Number:	7
Student Name:	Sakshi Morey
Roll No:	34

Title:

1. To write a program to demonstrate friend function in C++.
2. To write a program to demonstrate friend class in C++.

Learning Objective:

- Students will be able to implement friend function and friend classes in C++.

Learning Outcome:

- To understand how to use the private members using friend function and friend class.

Course Outcome:

ECL304.6	Percept the Utility and applicability of OOP

Theory:

- Explain in details about access specifiers: public, private and protected.

  Public: All the class members declared under the public specifier will be available to everyone. The data members and member functions declared as public can be accessed by other classes and functions too. The public members of a class can be accessed from anywhere in the program using the direct member access operator (.) with the object of that class.

  Private: The class members declared as *private* can be accessed only by the member functions inside the class. They are not allowed to be accessed directly by any object or function outside the class. Only the member functions or the friend functions are allowed to access the private data members of a class.

  Protected: Protected access modifier is similar to private access modifier in the sense that it can't be accessed outside of it's class unless with the help of friend class, the difference is that the class members declared as Protected can be accessed by any subclass(derived class) of that class as well.

Faculty: Ms. Deepali Kayande

- Explain about friend function and friend classes in C++.
  A **friend function** is a function that is specified outside a class but has the ability to access the class members' protected and private data. A friend can be a member's function, function template, or function, or a class or class template, in which case the entire class and all of its members are friends.

  A **friend class** can have access to the data members and functions of another class in which it is declared as a friend. They are used in situations where we want a certain class to have access to another class's private and protected members. Classes declared as friends to any another class will have all the member functions become friend functions to the friend class. Friend functions are used to work as a link between the classes.
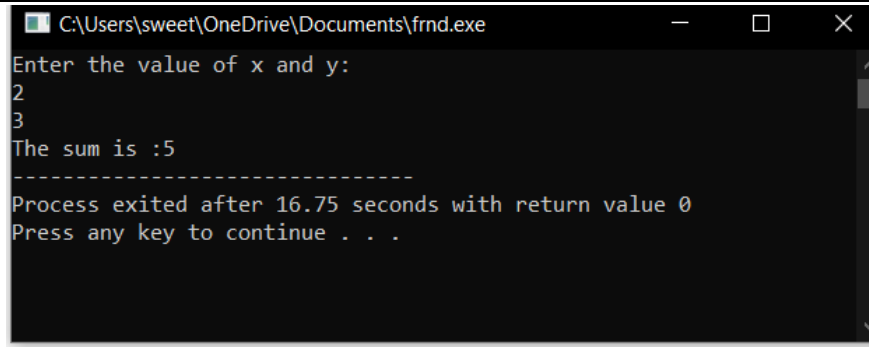
Algorithm 1:	1. Start the program.
	2. Declare the class name as temp with data members and member functions.
	3. The function input() is used to read the 2 inputs from the user.
	4. Declare the friend function mean add(temp &t) inside the class.
	5. Outside the class define the friend function.
	6. Return the value a + b.
	7. Stop the program.

Program 1:	```#include<iostream>```
	 ```using namespace std;```
	 ```class temp```
	 ```{```
	 ``` int a, b, add;```

```cpp
public:

void input()

{

cout << "Enter the value of x and y:";

cin >> a>>b;

}

friend void add(temp &t);

void display()

{

 cout << "The sum is :" << add;

}

};

void add(temp &t)

{

 t.add = t.a + t.b;

}

int main()

{

 temp t1;

 t1.input();

 add(t1);

 t1.display();

 return 0;

}
```

Output Screenshot 1:	
	C:\Users\sweet\OneDrive\Documents\frnd.exe — □ ✕ Enter the value of x and y: 2 3 The sum is :5 -------------------------------- Process exited after 16.75 seconds with return value 0 Press any key to continue . . .
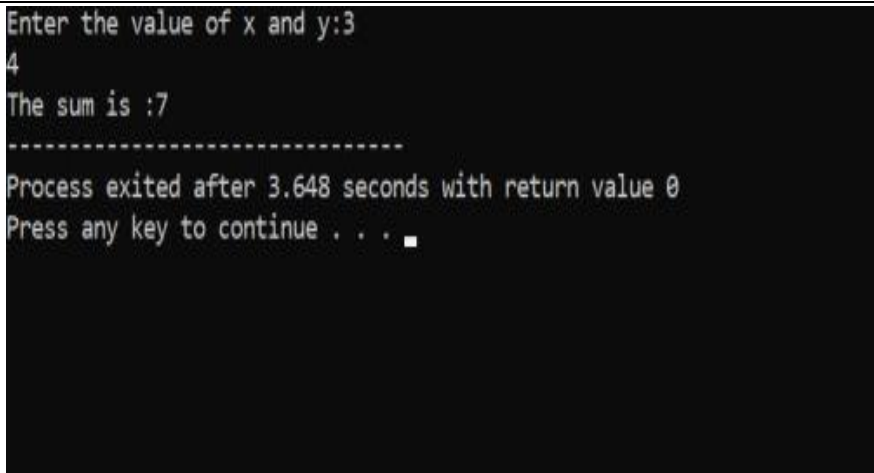
Algorithm 2:	1. Start the program.
	2. Declare the class name as First with data members and member functions.
	3. Declare the friend class Second.
	4. The function get_num to define a .
	5. Define the friend class Second
	6. The function get_num to define b.
	7. Declare the function get_sum to add two values inside the second class.
	8. Take the input of a and b inside the main function.
	9. Return the value a + b.
	10. Stop the program
| **Program 2:** | ```cpp
#include<iostream>
#include<stdio.h>
using namespace std;

class First
{
  int a;
  friend class Second;
  public:
  void get_num(int x)
  {
    a = x;
``` |

```cpp
        }
    };


    class Second
    {
      int b;
      public:
      void get_num(int y)
      {
         b = y;
      }


      void get_sum( First obj1)
       {
         int sum;
         sum = obj1.a + b;
         cout<<"Total: "<<sum;
       }
    };


    int main()
    {
      int a,b;
      First obj2;
      Second s;


      cout<<"Enter two number: ";
      cin>>a>>b;
```

	obj2.get_num(a);
	
	s.get_num(b);
	
	s.get_sum(obj2);
	
	return 0;
	
	}

Output Screenshot 2:	``` Enter the value of x and y:3 4 The sum is :7 ---------------------------------- Process exited after 3.648 seconds with return value 0 Press any key to continue . . . ```