**Object Oriented Programming**

**Lab report: 4**



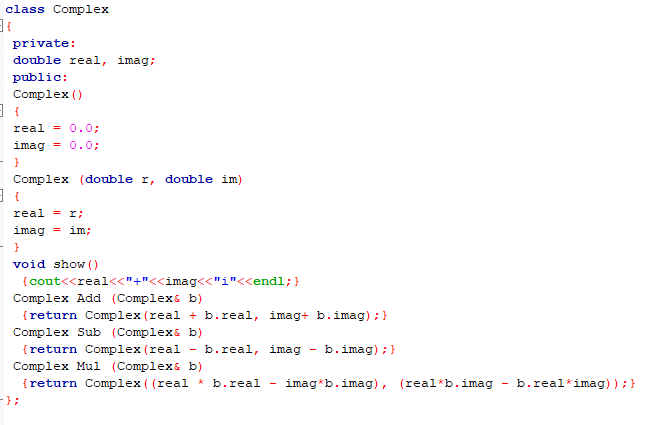
|  |  |
| --- | --- |
| Name | Ali Salman |
| Reg no | FA22-BCE-005 |
| Class | BCE- 4 |
| Instructor’s Name | Prof. Tayyab Rasul |

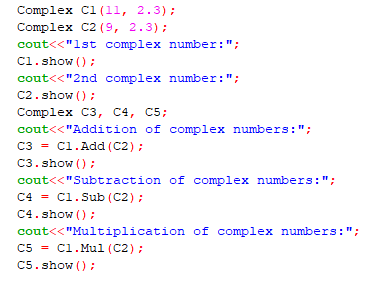
**Objects and Methods**

**Lab Tasks**

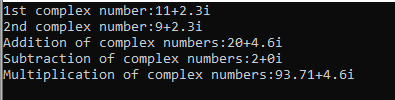
**5.1.** Code the example of complex class given above and include the functions for addition, subtraction and multiplication of objects of complex class and return the object containing result. Test all the functions in main.

Program:



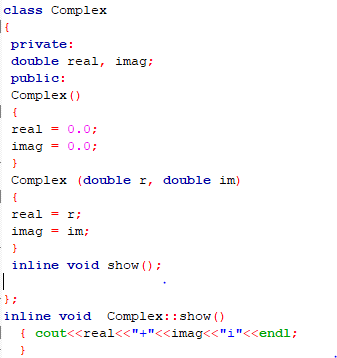


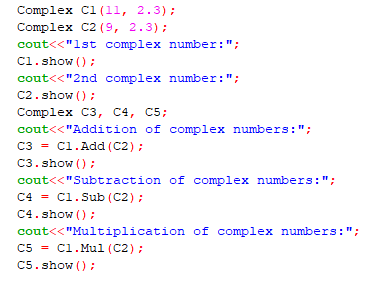
Output:



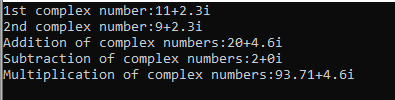
**5.2.** Modify the above task such that; define the member function show() outside the class and also define it as inline.

Program:



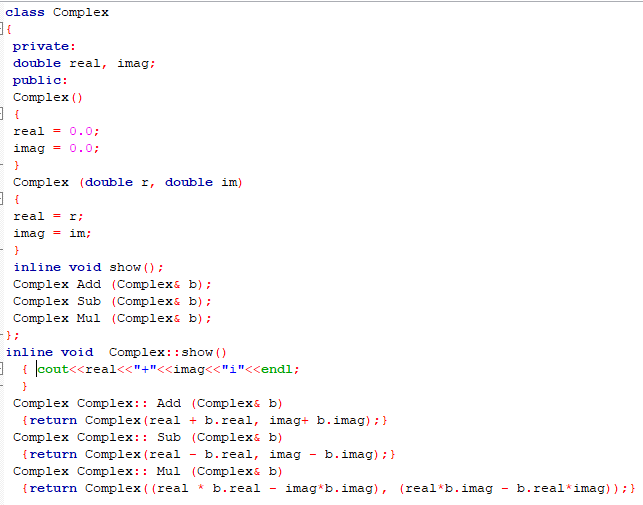


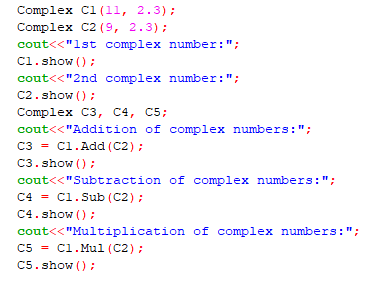
Output:



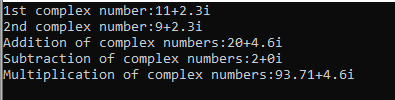
**5.3.** Modify the task (5.1) by defining all the member functions outside the class definition.

Program:



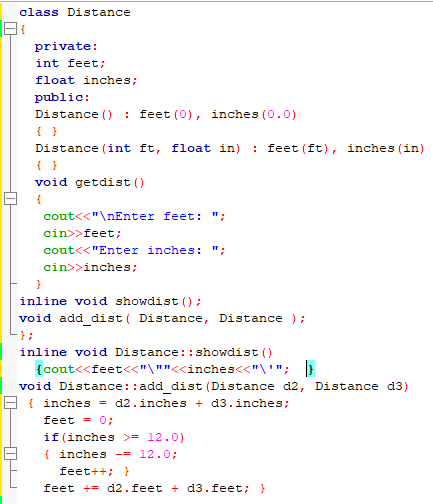


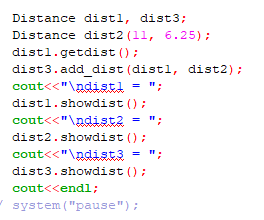
Output:



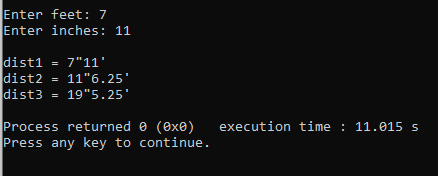
**5.4.** Test the distance class example given above.

Program:





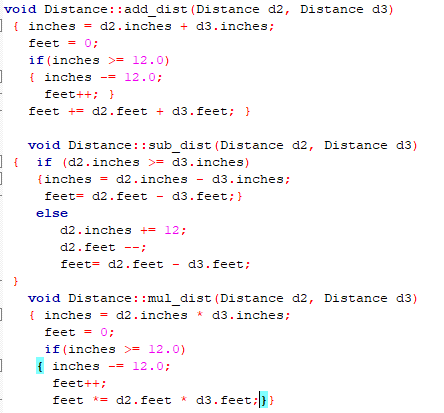
Output:

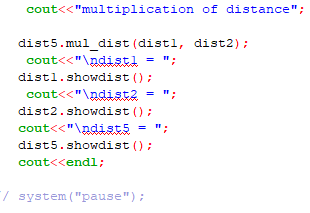


**Home Tasks**

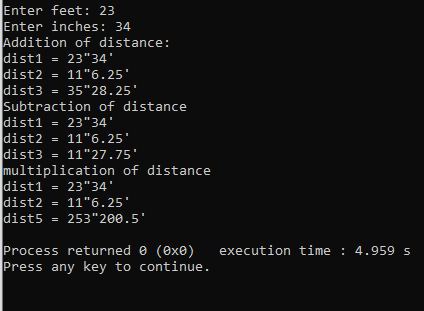
**6.1.** Modify the Distance class example of lab task (5.4) by including functions for subtraction and multiplication of distance class objects like addition.

Program:





Output:



Critical Analysis:

In this lab, we learn the basics of private and public specifiers. We learn about how functions are executed in public specifier and outside the class and how inline function is used. In order to minimize costly duplication and to allow other functions to access the same object as the calling function, C++ objects are typically supplied by reference. It is always possible for a member function to access the object for which it was called: the thing that's linked to it using the dot operator. We came to know that member functions with a small amount of code are stated inline. If you define a member function outside of its class declaration, the namespace scope that surrounds the class definition is where it must appear.