**TASK # 03:**

Question 1: Can a friend function be used to overload an operator that modifies the

invoking object?

Problem Statement:

Consider the += operator, which modifies the left-hand operand. Can a friend function be used

to overload this operator?

* If yes, how should it be implemented?
* If no, what alternative approach should be used?

**Ans: Yes, a friend function can be used to overload the += operator. A friend function has access to the private and protected members of the class, which allows it to modify the object.**

**Code:**

#include<iostream>

using namespace std;

class temp{

    private:

int a;

public:

temp(int a=0):a(a){}

friend temp& operator+=( temp& obj,const temp&obj2);

void display()const{

    cout<<"Value: "<<a<<endl;

}

};

temp&operator+=( temp& obj,const temp&obj2){

    obj.a+=obj2.a;

    return obj;

}

int main() {

   temp obj1(10), obj2(20);

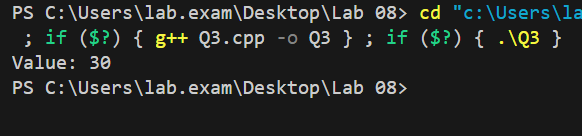
    obj1 += obj2;

    obj1.display();

    return 0;

}

**Output:**

****

Question 2: Is it possible to overload an operator using a friend function if one of the

operands is a primitive data type?

Problem Statement:

Suppose we want to overload the + operator to allow addition between an object and a primitive

type (e.g., object + int).

* Can a friend function handle this case?
* If yes, how would it be implemented?
* If no, what limitations exist?

**Ans: Yes friend function can handle this  
Code:**

#include <iostream>

using namespace std;

class temp

{

private:

    int a;

public:

    temp(int a = 0) : a(a) {}

    friend temp operator+(temp &obj, const int data);

    void display() const

    {

        cout << "Value: " << a << endl;

    }

};

temp operator+(temp &obj, const int data){

    return temp(obj.a+data);

}

int main(){

    temp obj1(10), obj2;

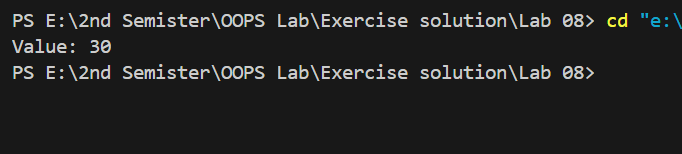
    obj2=obj1 + 20;

        obj2.display();

        return 0;

}

**Output:**



Question 3: Can a friend function access private and protected members of a class without

using an object of that class?

Problem Statement:

A friend function is granted access to private and protected members of a class.

* Does it always need an object to access these members?
* Can a friend function access them directly without an object?
* Under what conditions might it fail?

**Ans: Yes, a friend function always needs an object to access the private or protected members of a class.Since a friend function is not a member function, it does not have an implicit this pointer, which means it cannot access the private members unless an object of the class is passed to it.**