LAB_08

LAB_TASK_3

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?*

Justify your answer with supporting C++ code

ANS: YES, it can be used to overload += operator.

CODE:

```cpp
#include <iostream>
using namespace std;
class complex{
    private:
        int a;
    public:
        complex(int x =0):a(x){}
        friend complex operator +=(complex &c,const complex &c1);
        void show(){
            cout <<"A =     "<<a<<endl;
        }
};
complex operator+=(complex &c,const complex &c1){
    c.a+=c1.a;
    return c;
}
int main(){
    complex c(4),c1(4);
    c.show();
    c1.show();
    c+=c1;
    c.show();
}
```

OUTPUT:

```
PS C:\Users\OBAID\Desktop\FAST WORK\2nd Sem\OOP LAB\LAB_08\LAB_TASKS\TASK_3> ./a
A =     4
A =     4
A =     8
PS C:\Users\OBAID\Desktop\FAST WORK\2nd Sem\OOP LAB\LAB_08\LAB_TASKS\TASK_3>
```

QUESTION2:

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?

Justify your answer with supporting C++ code.

ANS: YES, it can be used to overload + operator.

CODE:

```
#include <iostream>
using namespace std;
class complex{
    private:
        int a;
    public:
        complex(int x =0):a(x){}
        friend complex operator +(complex &c,const int n);
        void show(){
            cout <<"A =     "<<a<<endl;
        }
};
complex operator+(complex &c,const int n){
    complex temp;
    temp.a=c.a+n;
    return temp;
}
int main(){
    complex c(4);
    c.show();
    c=c+20;
    c.show();
}
```

OUTPUT:

```
PS C:\Users\OBAID\Desktop\FAST WORK\2nd Sem\OOP LAB\LAB_08\LAB_TASKS\TASK_3> ./a
A =     4
A =     24
PS C:\Users\OBAID\Desktop\FAST WORK\2nd Sem\OOP LAB\LAB_08\LAB_TASKS\TASK_3>
```

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?

Justify your answer with supporting C++ code.

ANS: Friend Function can access private/protected data members of a class, but it can not access them without object. Since friend function is not a member function of class and it does not have "this" pointer, so it cannot access private/protected data members of class without the object.

CODE:

```
#include <iostream>
using namespace std;
class complex{
    private:
        int a;
    protected:
        int b;
    public:
        complex(int x =0,int y=0):a(x),b(y){}
        friend complex access(complex &c);
        void show(){

        }
};
complex access(complex &c){
    cout <<"A =      "<<c.a<<endl;
    cout <<"B =      "<<c.b<<endl;
    return 0;
}
int main(){
    complex c(4,5);
    access(c);
}
```

OUTPUT:

```
PS C:\Users\OBAID\Desktop\FAST WORK\2nd Sem\OOP LAB\LAB_08\LAB_TASKS\TASK_3> ./a
A =     4
B =     5
PS C:\Users\OBAID\Desktop\FAST WORK\2nd Sem\OOP LAB\LAB_08\LAB_TASKS\TASK_3>
```