**OOP LAB 8**

**TASK 3**

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

**Answer:**  
Yes, a friend function **can** be used to overload an operator like +=, but since it is **not a member** of the class, it cannot use this directly. Instead, it must take the left-hand object as a **reference** so it can modify it.

CPP CODE :

#include <iostream>

using namespace std;

class Number {

private:

    int value;

public:

    Number(int val) : value(val) {}

    friend Number& operator+=(Number& lhs, const Number& rhs);

    void display() {

        cout << "Value: " << value << endl;

    }

};

Number& operator+=(Number& lhs, const Number& rhs) {

    lhs.value += rhs.value;

    return lhs;

}

int main() {

    Number num1(10), num2(5);

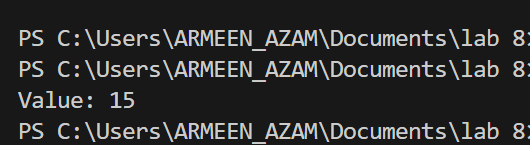
    num1 += num2;

    num1.display();

    return 0;

}

OUTPUT:



**Question 2: Can we overload an operator using a friend function if one operand is a primitive type?**

**Answer:**  
Yes, a friend function can be used to overload an operator when one operand is a primitive type. This is useful when we want to **add an object and a number** (e.g., object + int).

CPP CODE:

#include <iostream>

using namespace std;

class Number {

private:

    int value;

public:

    Number(int val) : value(val) {}

    friend Number operator+(const Number& obj, int num);

    void display() const {

        cout << "Value: " << value << endl;

    }

};

Number operator+(const Number& obj, int num) {

    return Number(obj.value + num);

}

int main() {

    Number num1(30);

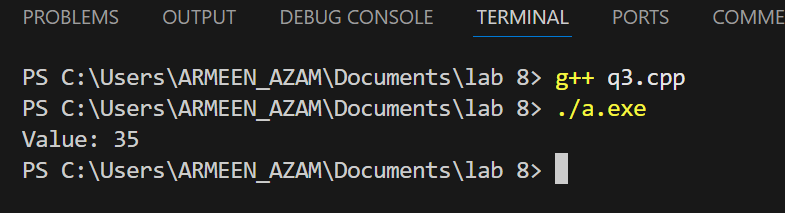
    Number result = num1 + 5;

    result.display();

    return 0;

}

OUTPUT:



**Question 3: Can a friend function access private members without using an object?**

**Answer:**  
No, a friend function **needs an object** to access private or protected members. It **cannot** access them without an object because private members **belong to an object, not the class itself**.

CPP CODE:

#include <iostream>

using namespace std;

class Number {

private:

    int value;

public:

    Number(int val) : value(val) {}

    friend void showValue(const Number& obj);

};

void showValue(const Number& obj) {

    cout << "Value: " << obj.value << endl;

}

int main() {

    Number num(20);

    showValue(num);

    return 0;

}

OUTPUT:

