**OOP LAB 8 – LAB TASK 3**

**Question 1:**

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

No, a friend function cannot be used to overload an operator like += that modifies the invoking object, because friend functions do not have an implicit this pointer. Instead,+= should be overloaded as a member function to modify the calling object.

**CODE:**

**#include <iostream>**

**using namespace std;**

**class Account {**

**double balance;**

**public:**

**Account(double b) : balance(b) {}**

**Account& operator+=(double amount) {**

**balance += amount;**

**return \*this;**

**}**

**void display() { cout << "Balance: Rs" << balance << endl; }**

**};**

**int main() {**

**Account acc(1000);**

**acc += 500;  // Adding 500 to balance**

**acc.display(); // Output: Balance: Rs1500**

**return 0;**

**}**

**Question 2:**

**Is it possible to overload an operator using a friend function if one of the operands is a primitive data type?**

Yes, a friend function can be used to overload an operator when one operand is a primitive type, such as (object + int) , but it does not work for (int + object) unless a separate overload is written. The friend function must take the object by reference or value and return a new object.

**CODE:**

**#include <iostream>**

**using namespace std;**

**class Account {**

**double balance;**

**public:**

**Account(double b) : balance(b) {}**

**friend Account operator+(const Account& acc, double amount) { // Overloading + operator**

**return Account(acc.balance + amount);**

**}**

**void display() { cout << "Balance: Rs" << balance << endl; }**

**};**

**int main() {**

**Account acc(2000);**

**Account acc2 = acc + 500; // Adding 500 to balance**

**acc2.display(); // Output: Balance: Rs2500**

**return 0;**

**}**

**Question 3:**

**Can a friend function access private and protected members of a class without using an object of that class?**

No, a friend function must have an object of the class to access private and protected members. A friend function does not have automatic access unless an instance of the class is provided.

**CODE:**

**#include <iostream>**

**using namespace std;**

**class Account {**

**double balance;**

**public:**

**Account(double b) : balance(b) {}**

**friend void showBalance(const Account& acc); // Friend function**

**};**

**void showBalance(const Account& acc) {**

**cout << "Balance: Rs" << acc.balance << endl; // Accessing private member**

**}**

**int main() {**

**Account acc(3000);**

**showBalance(acc); // Output: Balance: Rs3000**

**return 0;**

**}**