**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 the += operator, but it is not recommended because += modifies the invoking object, which is typically done through a member function. If a friend function is used, it must take a reference to the left-hand operand to modify it. However, the preferred approach is to implement operator+= as a member function.

#include <iostream>

using namespace std;

class Number {

int value;

public:

Number(int v) : value(v) {}

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;

}

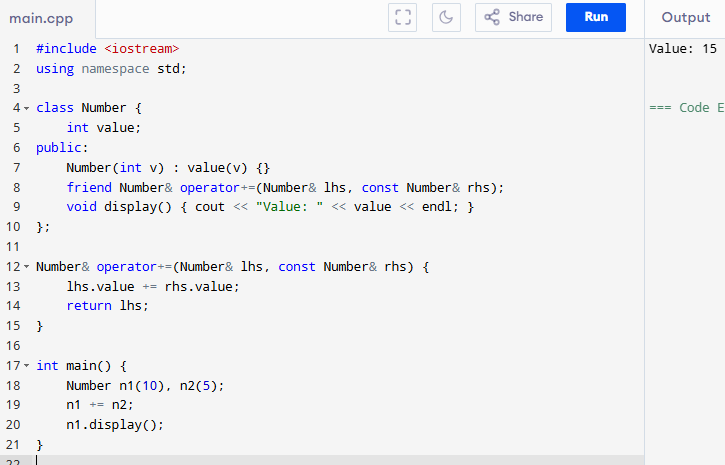
int main() {

Number n1(10), n2(5);

n1 += n2;

n1.display();

}



**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 data type. This is because friend functions allow overloading in cases where the left operand is not an object of the class (e.g., object + int or int + object). A member function cannot handle int + object because the left operand would need to be an instance of the class.

#include <iostream>

using namespace std;

class Number {

int value;

public:

Number(int v) : value(v) {}

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

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

void display() { cout << "Value: " << value << endl; }

};

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

return Number(obj.value + num);

}

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

return Number(num + obj.value);

}

int main() {

Number n1(10);

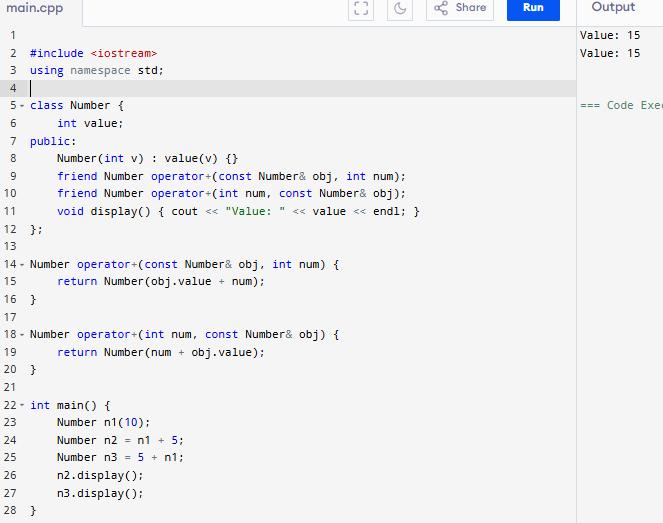
Number n2 = n1 + 5;

Number n3 = 5 + n1;

n2.display();

n3.display();

}



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

**Answer:** No, a friend function cannot access private or protected members of a class without an object. It always needs an object of the class to access these members because private and protected members belong to instances of the class, not the function itself. The function does not become a part of the class; it only has permission to access members through an object.

#include <iostream>

using namespace std;

class Number {

int value;

public:

Number(int v) : value(v) {}

friend void showValue(const Number& obj);

};

void showValue(const Number& obj)

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

int main() {

Number n(10);

showValue(n);

}

