1.
```
#include<iostream>

class Number
{
    int n;
public:
    Number(int x) :n(x) {}

    Number& operator++() { ++n; return *this; }

    Number& operator++(int) { n++; return *this; }

    friend Number& operator--(Number& o);
    friend Number& operator--(Number o, int);

    void display() { std::cout << "This Number is: " << n << std::endl; }
};
Number& operator--(Number& o) { --o.n; return o; }
Number& operator--(Number o, int) { o.n--; return o; }

int main()
{
    Number N1(10);
    ++ ++ ++N1;
    N1.display();
    N1++;
    N1.display();
    --N1;
    N1.display();
    N1------;
    N1.display();
    return 0;
}
```

```
This Number is: 13
This Number is: 14
This Number is: 13
This Number is: 13
```

2(1)

a=10

b=20

c=30

2(2)

S=ok

S=ok

2(3)

William 19 28000.998

Bob 17 1000.998

2(4)

d1=5.5

C2=5.5,0

3

```
#include <iostream>

class Calculator {
private:
    int count;

public:
    Calculator(int initial_count = 0) : count(initial_count) {}

    Calculator& operator++() {
        if (count < 65535) {
            ++count;
        }
        return *this;
    }

    Calculator operator++(int) {
        Calculator temp = *this;
        ++(*this);
        return temp;
    }

    Calculator& operator--() {
        if (count > 0) {
            --count;
        }
        return *this;
    }

    Calculator operator--(int) {
        Calculator temp = *this;
```

```
        --(*this);
        return temp;
    }

    Calculator operator+(const Calculator& other) const {
        int sum = count + other.count;
        return Calculator(sum % (65535 + 1));
    }

    Calculator operator-(const Calculator& other) const {
        if (count >= other.count) {
            return Calculator(count - other.count);
        }
        else {
            return Calculator((65535 + 1) + count - other.count);
        }
    }

    int getCount() const {
        return count;
    }
};

int main() {
    Calculator calc1(10), calc2(20);

    std::cout << "calc1: " << calc1.getCount() << std::endl;
    std::cout << "calc2: " << calc2.getCount() << std::endl;

    ++calc1;
    std::cout << "++calc1: " << calc1.getCount() << std::endl;

    calc1++;
    std::cout << "calc1++: " << calc1.getCount() << std::endl;

    --calc2;
    std::cout << "--calc2: " << calc2.getCount() << std::endl;

    calc2--;
    std::cout << "calc2--: " << calc2.getCount() << std::endl;

    Calculator sum = calc1 + calc2;
    std::cout << "calc1 + calc2: " << sum.getCount() << std::endl;
```

```
    Calculator diff = calc1 - calc2;
    std::cout << "calc1 - calc2: " << diff.getCount() << std::endl;

    return 0;
}
```

4

```
#include <iostream>
#include <cmath>
#include <iomanip>

class TwoCoor {
public:
    double x, y;

    TwoCoor(double x = 0, double y = 0) : x(x), y(y) {}

    TwoCoor operator+(const TwoCoor& other) const {
        return TwoCoor(this->x + other.x, this->y + other.y);
    }

    TwoCoor operator-(const TwoCoor& other) const {
        return TwoCoor(this->x - other.x, this->y - other.y);
    }

    double distance(const TwoCoor& other) const {
        double dx = this->x - other.x;
        double dy = this->y - other.y;
        return std::sqrt(dx * dx + dy * dy);
    }

    friend std::istream& operator>>(std::istream& is, TwoCoor& c);

    friend std::ostream& operator<<(std::ostream& os, const TwoCoor& c);
};

std::istream& operator>>(std::istream& is, TwoCoor& c) {
    is >> c.x >> c.y;
    return is;
}

std::ostream& operator<<(std::ostream& os, const TwoCoor& c) {
    os << "(" << std::setprecision(2) << std::fixed << c.x << ", " << c.y << ")";
    return os;
```

```
}

int main() {
    TwoCoor p1, p2;

    std::cout << "Enter coordinates for point 1: ";
    std::cin >> p1;
    std::cout << "Enter coordinates for point 2: ";
    std::cin >> p2;

    std::cout << "Point 1: " << p1 << std::endl;
    std::cout << "Point 2: " << p2 << std::endl;

    TwoCoor sum = p1 + p2;
    std::cout << "Sum: " << sum << std::endl;

    TwoCoor diff = p1 - p2;
    std::cout << "Difference: " << diff << std::endl;

    double distance = p1.distance(p2);
    std::cout << "Distance: " << distance << std::endl;

    return 0;
}
```