Q1. Given a circle with radius ***r***. Write a **C++** code to create the following:

Solution

```cpp
#include <iostream>
using namespace std;

#define ll long long
#define ld long double

class Circle{
    private:
        ld _r;
    public:
        // Constructor
        Circle(): _r(0.0){}
        Circle(ld radius): _r(radius){}

        // Overloading
        Circle operator++(){
            return Circle(2 * _r);
        }

        // Functionality
        void display(){
            printf("Circle(radius = %5.2Lf)\n", _r);
        }
};


int main(){
    Circle Cir1, Cir2(15), Cir3 = ++Cir2;

    // test
    Cir1.display();
    Cir2.display();
    Cir3.display();

    return 0;
}

/* output
    PS GU\OOP\LEC\Assignments\#7\code> c++ .\main.cpp -o main.exe; .\main.exe
    Circle(radius =  0.00)
    Circle(radius = 15.00)
    Circle(radius = 30.00)
*/
```

Q2. A vector in the polar coordinates has two parameters radius **r** and angle **Theta**. Write a **C++** code to create the following:

Solution

```cpp
#include <iostream>
using namespace std;

#define ll long long
#define ld long double

class Vector{
    private:
        ld _r, _theta;
    public:
        // Constructor
        Vector(): _r(1), _theta(0){}
        Vector(ld radius, ld theta): _r(radius), _theta(theta){}

        // Overloading
        bool operator==(const Vector& C){
            return (_r == C._r && _theta == C._theta);
        }

        // Functionality
        void display(){
            printf("Vector(radius = %8.2Lf, theta = %8.2Lf)\n", _r, _theta);
        }
};


int main(){
    Vector Vec1, Vec2(15, 180), Vec3;

    // test
    Vec1.display();
    Vec2.display();
    Vec3.display();

    cout << "Vec1 == Vec3: " << (Vec1 == Vec3) << endl; // yes: true > 1
    cout << "Vec1 == Vec2: " << (Vec1 == Vec2) << endl; // no: false > 0

    return 0;
}

/* output
    PS GU\OOP\LEC\Assignments\#7\code> c++ .\main.cpp -o main.exe; .\main.exe
    Vector(radius =     1.00, theta =     0.00)
    Vector(radius =    15.00, theta =   180.00)
    Vector(radius =     1.00, theta =     0.00)
    Vec1 == Vec3: 1
    Vec1 == Vec2: 0
*/
```

Q3. A complex variable has two parts, the real part and imaginary part. Write a **C++** code to create the following:

Solution

```cpp
#include <iostream>
using namespace std;

#define ll long long
#define ld long double

class Complx{
    private:
        ll int _real, _imag;
    public:
        // Constructor
        Complx(): _real(0), _imag(0){}
        Complx(ll int real, ll int imag): _real(real), _imag(imag){}

        // Overloading
        Complx operator+(const Complx& C){
            return Complx(_real + C._real, _imag + C._imag);
        }

        // Functionality
        void display(){
            printf("Complx(real = %lli, imaginary = %lli)\n", _real, _imag);
        }
};

int main(){
    Complx C1, C2(15, 12), C3 = C1 + C2;

    // test
    C1.display();
    C2.display();
    C3.display();

    return 0;
}

/* output
    PS GU\OOP\LEC\Assignments\#7\code> c++ .\main.cpp -o main.exe; .\main.exe
    Complx(real = 0, imaginary = 0)
    Complx(real = 15, imaginary = 12)
    Complx(real = 15, imaginary = 12)
*/
```

Q4. A vector has two parameters magnitude and direction. Write a **C++** code to create the following:

Solution

```cpp
#include <iostream>
using namespace std;

#define ll long long
#define ld long double

class Vector{
    private:
        ll int _mag, _direction;
    public:
        // Constructor
        Vector(): _mag(0), _direction(1){}
        Vector(ll int mag, ll int direction): _mag(mag), _direction(direction){}

        // Overloading
        Vector operator-(){
            return Vector(_mag, -1 * _direction);
        }

        // Functionality
        void display(){
            printf("Vector(mag = %lli, dire = %lli)\n", _mag, _direction);
        }
};

int main(){
    Vector C1, C2(15, 12), C3 = -C2;

    // test
    C1.display();
    C2.display();
    C3.display();

    return 0;
}

/* output
    PS GU\OOP\LEC\Assignments\#7\code> c++ .\main.cpp -o main.exe; .\main.exe
    Vector(mag = 0, dire = 1)
    Vector(mag = 15, dire = 12)
    Vector(mag = 15, dire = -12)
*/
```