Assignment 04

4.1 Design a class Polar which describes a point in the plane using polar coordinates radius and angle. A point in polar coordinates is shown in the figure below. Use the overload + operator to add two objects of Polar.

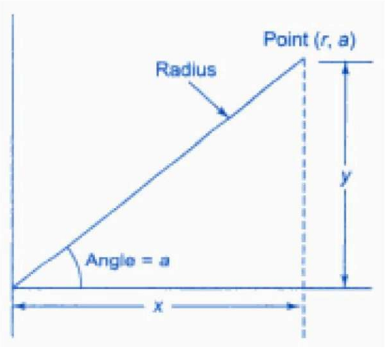

Figure 1: Polar coordinates of a point

Note that we cannot add polar values of two points directly. This requires first the conversion of points into rectangular coordinates, then adding the respective rectangular coordinates and finally converting the result back into polar coordinates. You need to use the following trigonometric formula:

- x = r * cos(a);
- y = r * sin(a);
- a = atan(y/x); //arc tangent
- r = sqrt(x*x + y*y);

```
#include <iostream>
#include <cmath>
#define PI 3.14159265
using namespace std;

class Polar {
    float r, a; // r = radius, a = angle in degrees
public:
    Polar() : r(0), a(0) {}
    Polar(float radius, float angle) : r(radius), a(angle) {}

    Polar operator+(Polar p) {
```

```cpp
        // Convert both to rectangular coordinates
        float x1 = r * cos(a * PI / 180);
        float y1 = r * sin(a * PI / 180);
        float x2 = p.r * cos(p.a * PI / 180);
        float y2 = p.r * sin(p.a * PI / 180);

        // Add the coordinates
        float x = x1 + x2;
        float y = y1 + y2;

        // Convert back to polar
        float result_r = sqrt(x * x + y * y);
        float result_a = atan2(y, x) * 180 / PI; // angle in degrees

        return Polar(result_r, result_a);
    }

    void display() {
        cout << "Radius = " << r << ", Angle = " << a << " degrees" <<
endl;
    }
};

int main() {
    Polar p1(5, 45), p2(8, 90);
    Polar p3 = p1 + p2;

    cout << "p1: ";
    p1.display();
    cout << "p2: ";
    p2.display();
    cout << "p1 + p2: ";
    p3.display();

    return 0;
}
```

Sample Output:-

```
p1: Radius = 5, Angle = 45 degrees
p2: Radius = 8, Angle = 90 degrees
p1 + p2: Radius = 11.1803, Angle = 69.2952 degrees
```

4.2 Write a program to show how the increment operator (++) is overloaded.

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

class Counter {
    int value;
public:
    Counter(int v = 0) : value(v) {}

    // Prefix increment
    Counter& operator++() {
        ++value;
        return *this;
    }

    // Postfix increment
    Counter operator++(int) {
        Counter temp = *this;
        value++;
        return temp;
    }

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

int main() {
    Counter c(5);
    cout << "Initial: ";
    c.display();

    ++c;
    cout << "After prefix ++: ";
    c.display();

    c++;
    cout << "After postfix ++: ";
    c.display();

    return 0;
}
```

Sample Output:-

```
Initial: Value: 5
After prefix ++: Value: 6
After postfix ++: Value: 7
```

4.3 Crate a class FLOAT that contains one float data member. Overload all the four arithmetic operators so that they operate on the objects of FLOAT.

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

class FLOAT {
    float value;
public:
    FLOAT(float v = 0.0) : value(v) {}

    FLOAT operator+(const FLOAT &f) const {
        return FLOAT(value + f.value);
    }
    FLOAT operator-(const FLOAT &f) const {
        return FLOAT(value - f.value);
    }
    FLOAT operator*(const FLOAT &f) const {
        return FLOAT(value * f.value);
    }
    FLOAT operator/(const FLOAT &f) const {
        if (f.value == 0) {
            cout << "Division by zero!" << endl;
            return FLOAT();
        }
        return FLOAT(value / f.value);
    }
    void display() const {
        cout << value << endl;
    }
};

int main() {
    FLOAT a(10.6), b(5.3), c;
    cout << "a = 10.6, b = 5.3" << endl;

    c = a + b;
    cout << "a + b = "; c.display();

    c = a - b;
    cout << "a - b = "; c.display();

    c = a * b;
    cout << "a * b = "; c.display();

    c = a / b;
    cout << "a / b = "; c.display();

    return 0;
}
```

Sample Output:-

```
a = 10.6, b = 5.3
a + b = 15.9
a - b = 5.3
a * b = 56.18
a / b = 2
```