```
//Devin Hardy
//CS372

#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <cmath>

using namespace std;

class point
{
private:
    float xCoord;
    float yCoord;

public:
    // Constructor
    point(float x = 0, float y = 0);
    // Method
    void shift(float shiftX, float shiftY);
    void rotate90();
    int rotations_needed(point p);
    float getX();
    float getY();
    double distance(point &p1, point &p2);
    point middle(const point &p1, const point &p2);
    // Manipulate point
    void Translation(float x, float y);
    void Scaling(float mX, float mY);
    void Rotation(double degree);
    void Shearing(float x, float y);
    // Overload
    bool operator==(point &p1);
    bool operator!=(point &p1);
    point operator+(point &p1);
    point operator-(point &p1);
    friend ostream& operator<<(ostream &out, const point &p);

};

//Constructor
    point::point(float x, float y)
    {
        xCoord = x;
        yCoord = y;
    }
// Method
    void point::shift(float shiftX, float shiftY)
    {
```

```
    xCoord = xCoord + shiftX;
    yCoord = yCoord + shiftY;
}
// Rotate Clockwise 90 degrees
void point::rotate90()
{
    float tempX, tempY;
    tempX = xCoord;
    tempY = yCoord;
    xCoord = tempY;
    yCoord = -1 * tempX;
}

int point::rotations_needed(point p)
{
    int rotNeed = 0;
    while(p.getX() < 0.0 || p.getY() < 0.0)
    {
        rotNeed ++;
        p.rotate90();
    }
    return rotNeed;
}

float point::getX()
```

```
{
    return xCoord;
}

float point::getY()
{
    return yCoord;
}

double point::distance(point &p1, point &p2)
{
    double a, b;
    a = p1.getX() - p2.getX();
    b = p1.getX() - p2.getX();
    return(a*a + b*b);
}

point point::middle(const point &p1, const point &p2)
{
    float mpx, mpy;
    mpx = (p1.xCoord + p2.xCoord) / 2;
    mpy = (p1.yCoord + p2.yCoord) / 2;
    point mid(mpx, mpy);
    return mid;
}
```

```
// Manipulate Point
void point::Translation(float x, float y)
{
    xCoord = xCoord + x;
    yCoord = yCoord + y;
    return;
}

void point::Scaling(float mX, float mY)
{
    xCoord = xCoord * mX;
    yCoord = yCoord * mY;
    return;
}

void point::Rotation(double degree)
{
    float x, y;
    x = xCoord;
    y = yCoord;
    xCoord = x*(cos(degree)) - y*(sin(degree));
    yCoord = x*(sin(degree)) + y*(cos(degree));
}
```

```
void point::Shearing(float x, float y)
{
    xCoord = xCoord + x;
    yCoord = yCoord + y;
    return;
}


bool point::operator==(point &p1)
{
    return (xCoord == p1.getX() && yCoord == p1.getY());
}

bool point::operator!=(point &p1)
{
    return !(xCoord == p1.getX() && yCoord == p1.getY());
}

bool operator<(point &p1, point &p2)
{
    point Origin;
    double dist1;
    double dist2;
    dist1 = Origin.distance(Origin, p1);
    dist2 = Origin.distance(Origin, p2);
    if(dist1 < dist2)
        return 1;
    else
        return 0;
}

bool operator>(point &p1, point &p2)
{
    point Origin;
    double dist1;
    double dist2;
    dist1 = Origin.distance(Origin, p1);
    dist2 = Origin.distance(Origin, p2);
    if(dist1 > dist2)
        return 1;
    else
        return 0;
}

bool operator>=(point &p1, point &p2)
{
    point Origin;
    double dist1;
    double dist2;
    dist1 = Origin.distance(Origin, p1);
    dist2 = Origin.distance(Origin, p2);
    if(dist1 >= dist2)
        return 1;
    else
        return 0;
}

point point::operator+(point &p1)
{
    float sumX, sumY;
    sumX = xCoord + p1.getX();
    sumY = yCoord + p1.getY();
    point sumCoord(sumX, sumY);
    return sumCoord;
}

point point::operator-(point &p1)
{
    float difX, difY;
    difX = xCoord - p1.getX();
    difY = yCoord - p1.getY();
    point sumCoord(difX, difY);
    return sumCoord;
}
```

```
    ostream& operator<<(ostream &out, const point &p)
    {
        out << '(' << p.xCoord << ',' << p.yCoord << ')';
        return out;
    }

int main()
{
    point Point1(12, -18);
    point Point2(10, 10);
    cout << Point1 << endl;
    // Shift Test
    cout << "Shift Test" << endl;
    cout << "Shift x by -2 and y by 8" << endl;
    Point1.shift(-2, 8);
    cout << Point1 << endl;
    // Rotate 90 Test
    cout << "Rotate 90" << endl;
    Point1.rotate90();
    cout << Point1 << endl;
    // Rotates needed
    cout << "Rotation Needed" << endl;
    int needed;
    needed = Point1.rotations_needed(Point1);
    cout << needed << " rotations are needed." << endl;
    // Distance Test
    cout << "Distance Test" << endl;
    double dist;
    dist = Point1.distance(Point1, Point2);
    cout << "Distance between\n" << Point1 << " and " << Point2 << "
is\n" << dist << endl;
    // Middle Test
    cout << "Middle Test" << endl;
    point Middle;
    Middle = Point1.middle(Point1, Point2);
    cout << Point1 << " " << Middle << " " << Point2 << endl << endl;

    // 4 Points Test
    point boxP1(2,2);
    point boxP2(2,5);
    point boxP3(4,5);
    point boxP4(4,2);
    cout << boxP1 << ' ' << boxP2 << ' ' << boxP3 << ' ' << boxP4 <<
endl;
    boxP1.Translation(2,1);
    boxP2.Translation(2,1);
    boxP3.Translation(2,1);
    boxP4.Translation(2,1);
    cout << "Translation" << endl;
    cout << boxP1 << ' ' << boxP2 << ' ' << boxP3 << ' ' << boxP4 <<
endl;
```

```
    boxP1.Scaling(2, 0.5);
    boxP2.Scaling(2, 0.5);
    boxP3.Scaling(2, 0.5);
    boxP4.Scaling(2, 0.5);
    cout << "Scaling" << endl;
    cout << boxP1 << ' ' << boxP2 << ' ' << boxP3 << ' ' << boxP4 <<
endl;

    point boxP5(2,2);
    point boxP6(2,5);
    point boxP7(4,5);
    point boxP8(4,2);

    boxP5.Rotation(30);
    boxP6.Rotation(30);
    boxP7.Rotation(30);
    boxP8.Rotation(30);
    cout << "Rotation 30" << endl;
    cout << boxP5 << ' ' << boxP6 << ' ' << boxP7 << ' ' << boxP8 <<
endl;

    boxP5.Rotation(60);
    boxP6.Rotation(60);
    boxP7.Rotation(60);
    boxP8.Rotation(60);
    cout << "Rotation 60" << endl;
    cout << boxP5 << ' ' << boxP6 << ' ' << boxP7 << ' ' << boxP8 <<
endl;

    point boxP9(2,5);
    point boxP10(2,5);
    point boxP11(4,5);
    point boxP12(4,2);

    boxP9.Shearing(1.5, 0);
    boxP10.Shearing(1.5, 0);
    boxP11.Shearing(1.5, 0);
    boxP12.Shearing(1.5, 0);
    cout << "Shearing X" << endl;
    cout << boxP9 << ' ' << boxP10 << ' ' << boxP11 << ' ' << boxP12
<< endl;

    point boxP13(2,2);
    point boxP14(2,5);
    point boxP15(4,5);
    point boxP16(4,2);

    boxP13.Shearing(0, 1.7);
    boxP14.Shearing(0, 1.7);
    boxP15.Shearing(0, 1.7);
    boxP16.Shearing(0, 1.7);
    cout << "Shearing Y" << endl;
    cout << boxP13 << ' ' << boxP14 << ' ' << boxP15 << ' ' << boxP16
<< endl;

    // Overload Test
    point Point3(3,4);
    point Point4(3,4);
    point addP;
    cout << " Equal to Test" << endl;
    if(Point1 == Point2)
        cout << Point1 << " is equal to " << Point2 << endl;
    else
        cout << Point1 << " is not equal to " << Point2 << endl;
    if(Point3 == Point4)
        cout << Point3 << " is equal to " << Point4 << endl;
    else
        cout << Point3 << " is not equal to " << Point4 << endl;
    cout << endl;
    cout << "Not Equal to Test" << endl;
    if(Point1 != Point2)
        cout << Point1 << " is not equal to " << Point2 << endl;
    else
        cout << Point1 << " is equal to " << Point2 << endl;
    if(Point3 != Point4)
        cout << Point3 << " is not equal to " << Point4 << endl;
    else
        cout << Point3 << " is equal to " << Point4 << endl;
    cout << endl;

    cout << "Add two Points" << endl;
    addP = Point3 + Point4;
    cout << Point3 << " + " << Point4 << " = " << addP << endl;
    cout << endl;
    cout << "Subtract two Points" << endl;
    addP = Point1 + Point3;
    cout << Point1 << " - " << Point3 << " = " << addP << endl;
    cout << endl;

    cout << "Greater than Test" << endl;
    if(Point1 > Point3)
        cout << Point1 << " is greater than " << Point3 << endl;
    else
        cout << Point1 << " is less than " << Point3 << endl;
    cout << " from the origin (0,0)" << endl;
    if(Point3 > Point2)
        cout << Point3 << " is greater than " << Point2 << endl;
    else
        cout << Point3 << " is less than " << Point2 << endl;
    cout << " from the origin (0,0)" << endl;
    cout << endl;

    cout << "Less than Test" << endl;
```

```
    if(Point3 < Point2)
        cout << Point3 << " is less than " << Point2 << endl;
    else
        cout << Point3 << " is greater than " << Point2 << endl;
    cout << " from the origin (0,0)" << endl;
    if(Point3 < Point1)
        cout << Point3 << " is less than " << Point1 << endl;
    else
        cout << Point3 << " is greater than " << Point1 << endl;
    cout << " from the origin (0,0)" << endl;
    cout << endl;

    cout << "Greater than or equal too" << endl;
    Point3.rotate90();
    if(Point4 >= Point3)
        cout << Point4 << " is greater than or equal too " << Point3 <<
endl;
    else
        cout << Point4 << " is less than " << Point3 << endl;
    cout << " from the origin (0,0)" << endl;
    if(Point1 >= Point3)
        cout << Point4 << " is greater than or equal too " << Point1 <<
endl;
    else
        cout << Point4 << " is less than " << Point1 << endl;
    cout << " from the origin (0,0)" << endl;
    cout << endl;


    return 0;
}
```

```
(12,-18)
Shift Test
Shift x by -2 and y by 8
(10,-10)
Rotate 90
(-10,-10)
Rotation Needed
2 rotations are needed.
Distance Test
Distance between
(-10,-10) and (10,10) is
800
Middle Test
(-10,-10) (0,0) (10,10)

(2,2) (2,5) (4,5) (4,2)
Translation
(4,3) (4,6) (6,6) (6,3)
Scaling
(8,1.5) (8,3) (12,3) (12,1.5)
Rotation 30
(2.28457,-1.66756) (5.24866,-1.20481) (5.55716,-3.18087) (2.59307,-3.64362)
Rotation 60
(-2.68414,0.891846) (-5.36613,-0.452375) (-6.26228,1.33562) (-3.58029,2.67984)
Shearing X
(3.5,5) (3.5,5) (5.5,5) (5.5,2)
Shearing Y
(2,3.7) (2,6.7) (4,6.7) (4,3.7)
 Equal to Test
(-10,-10) is not equal to (10,10)
(3,4) is equal to (3,4)

Not Equal to Test
(-10,-10) is not equal to (10,10)
(3,4) is equal to (3,4)

Add two Points
(3,4) + (3,4) = (6,8)

Subtract two Points
(-10,-10) - (3,4) = (-7,-6)

Greater than Test
(-10,-10) is greater than (3,4)
 from the origin (0,0)
(3,4) is less than (10,10)
 from the origin (0,0)

Less than Test
(3,4) is less than (10,10)
 from the origin (0,0)
(3,4) is less than (-10,-10)
 from the origin (0,0)

Greater than or equal too
(3,4) is less than (4,-3)
 from the origin (0,0)
(3,4) is greater than or equal too (-10,-10)
 from the origin (0,0)
```