Course: ENSF 614–Fall2021

Lab #: Lab 5

Student Names: Graydon Hall, Jared Kraus

Submission Date: 2021-10-25

Exercise B

Program Output (copied from terminal)

```
Expected to dispaly the distance between m and n is: 3
The distance between m and n is: 3
Expected second version of the distance function also print: 3
The distance between m and n is again: 3

Testing Functions in class Square:

Square Name: SQUARE - S
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:          12.00
Area:           144.00
Perimeter:       48.00

Testing Functions in class Rectangle:
Rectangle Name: RECTANGLE A
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:          12.00
Side b:          15.00
Area:           180.00
Perimeter:       54.00

Rectangle Name: RECTANGLE B
X-coordinate:    16.00
Y-coordinate:     7.00
Side a:           8.00
Side b:           9.00
Area:            72.00
Perimeter:       34.00

Distance between square a, and b is: 11.00

Rectangle Name: RECTANGLE A
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:          12.00
Side b:          15.00
Area:           180.00
Perimeter:       54.00

Testing assignment operator in class Rectangle:

Rectangle Name: RECTANGLE rec2
```

```
X-coordinate:     3.00
Y-coordinate:     4.00
Side a:          11.00
Side b:           7.00
Area:            77.00
Perimeter:       36.00

Expected to display the following values for objec rec2:
Rectangle Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Side a: 12
Side b: 15
Area: 180
Perimeter: 54

If it doesn't there is a problem with your assignment operator.


Rectangle Name: RECTANGLE A
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:          12.00
Side b:          15.00
Area:           180.00
Perimeter:       54.00

Testing copy constructor in class Rectangle:

Rectangle Name: RECTANGLE A
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:         100.00
Side b:         200.00
Area:         20000.00
Perimeter:      600.00

Expected to display the following values for objec rec2:
Rectangle Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Side a: 100
Side b: 200
Area: 20000
Perimeter: 600

If it doesn't there is a problem with your assignment operator.
```

```
Rectangle Name: RECTANGLE A
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:         100.00
Side b:         200.00
Area:         20000.00
Perimeter:      600.00

Testing array of pointers and polymorphism:

Square Name: SQUARE - S
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:          12.00
Area:           144.00
Perimeter:       48.00

Rectangle Name: RECTANGLE B
X-coordinate:    16.00
Y-coordinate:     7.00
Side a:           8.00
Side b:           9.00
Area:            72.00
Perimeter:       34.00

Rectangle Name: RECTANGLE A
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:          12.00
Side b:          15.00
Area:           180.00
Perimeter:       54.00

Rectangle Name: RECTANGLE A
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:         100.00
Side b:         200.00
Area:         20000.00
Perimeter:      600.00
```

Code files:

```cpp
/* File Name: GraphicsWorld.cpp
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#include "GraphicsWorld.h"
#include "Point.h"
#include "Shape.h"
#include "Square.h"
#include <iostream>
using namespace std;
#include "Rectangle.h"
#include "Circle.h"
#include "CurveCut.h"

void GraphicsWorld::run(){
    #if 1 // Change 0 to 1 to test Point
        Point m (6, 8);
        Point n (6,8);
        n.setx(9);
        cout << "\nExpected to dispaly the distance between m and n is: 3";
        cout << "\nThe distance between m and n is: " << m.distance(n);
        cout << "\nExpected second version of the distance function also print:
3";
        cout << "\nThe distance between m and n is again: "
        << Point::distance(m, n);
    #endif // end of block to test Point
    #if 1 // Change 0 to 1 to test Square
        cout << "\n\nTesting Functions in class Square:" <<endl;
        Square s(5, 7, 12, "SQUARE - S");
        s.display();
    #endif // end of block to test Square
    #if 1 // Change 0 to 1 to test Rectangle
        cout << "\nTesting Functions in class Rectangle:";
        Rectangle a(5, 7, 12, 15, "RECTANGLE A");
        a.display();
        Rectangle b(16 , 7, 8, 9, "RECTANGLE B");
        b.display();
        double d = a.distance(b);
        cout <<"\nDistance between square a, and b is: " << d << endl;
        Rectangle rec1 = a;
        rec1.display();
        cout << "\nTesting assignment operator in class Rectangle:" <<endl;
        Rectangle rec2 (3, 4, 11, 7, "RECTANGLE rec2");
        rec2.display();
        rec2 = a;
        a.set_side_b(200);
        a.set_side_a(100);
        cout << "\nExpected to display the following values for objec rec2: "
<< endl;
        cout << "Rectangle Name: RECTANGLE A\n" << "X-coordinate: 5\n" << "Y-
coordinate: 7\n"
        << "Side a: 12\n" << "Side b: 15\n" << "Area: 180\n" << "Perimeter:
54\n" ;
        cout << "\nIf it doesn't there is a problem with your assignment
operator.\n" << endl;
        rec2.display();
        cout << "\nTesting copy constructor in class Rectangle:" <<endl;
        Rectangle rec3 (a);
        rec3.display();
        a.set_side_b(300);
        a.set_side_a(400);
        cout << "\nExpected to display the following values for objec rec2: "
<< endl;
        cout << "Rectangle Name: RECTANGLE A\n" << "X-coordinate: 5\n" << "Y-
coordinate: 7\n"
        << "Side a: 100\n" << "Side b: 200\n" << "Area: 20000\n" << "Perimeter:
600\n" ;
        cout << "\nIf it doesn't there is a problem with your assignment
operator.\n" << endl;
        rec3.display();
    #endif // end of block to test Rectangle
    #if 1 // Change 0 to 1 to test using array of pointer and polymorphism
        cout << "\nTesting array of pointers and polymorphism:" <<endl;
        Shape* sh1[4];
        sh1[0] = &s;
        sh1[1] = &b;
        sh1 [2] = &rec1;
        sh1 [3] = &rec3;
        sh1 [0]->display();
        sh1 [1]->display();
        sh1 [2]->display();
        sh1 [3]->display();
    #endif // end of block to test array of pointer and polymorphism
```

```cpp
int main(){
    GraphicsWorld x;
    x.run();
    return 0;
}
```

```cpp
/* File Name: GraphicsWorld.h
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#ifndef GRAPHICS_WORLD
#define GRAPHICS_WORLD

class GraphicsWorld{
public:
    void run();

};


#endif
```

```cpp
/* File Name: Point.cpp
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#include "Point.h"
#include <iostream>
using namespace std;
#include <math.h>
#include <cmath>
#include <iomanip>

int Point::point_counter=0;
int Point::id_counter=1000;

Point::Point(double x, double y){
    xcoordinate = x;
    ycoordinate = y;
    point_counter++;
    id_counter++;
    pointID = id_counter;
}

void Point::display(){
    cout << fixed;
    cout << setprecision(2);
    cout << "\nX-coordinate: " << setw(9) << xcoordinate << endl;
    cout << "Y-coordinate: " << setw(9) << ycoordinate << endl;
}

double Point::distance(const Point& m, const Point& n){
    // Pass m and n by reference to uneccesary destructor call for them
    return sqrt(pow(abs(m.getx() - n.getx()),2)+pow(abs(m.gety() -
n.gety()),2));
}

double Point::distance(const Point &p){
    // Pass p by reference to uneccesary destructor call for it
    return sqrt(pow(abs(getx() - p.getx()),2)+pow(abs(gety() - p.gety()),2));
}

Point::~Point(){
    point_counter --;
}
```

```




/* File Name: Point.h
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#ifndef POINT
#define POINT

class Point{
private:
    double xcoordinate;
    double ycoordinate;
    int pointID;
    static int point_counter;  // counter for # of points created
    static int id_counter; // assign IDs to each point
public:
    Point(double x, double y);
    ~Point();
    static double distance(const Point& m, const Point& n);
    double distance(const Point &p);
    static int counter(){return point_counter;}
    void display();

    void setx(double value){xcoordinate=value;}
    void sety(double value){ycoordinate=value;}
    double getx() const{return xcoordinate;}
    double gety() const{return ycoordinate;}
    int getID() const{return pointID;}

};


#endif
```

```
/* File Name: Rectangle.cpp
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

using namespace std;
#include <iostream>
#include <math.h>
#include <cmath>
#include <iomanip>
#include <string.h>

#include "Square.h"
#include "Rectangle.h"
#include "Shape.h"
#include "Point.h"
```

```
Rectangle::Rectangle(double x, double y, double a, double b, const char* name):
    Square(x, y, a, name), Shape(x, y, name)
{
    side_b = b;
}

void Rectangle::display(){
    cout << fixed;
    cout << setprecision(2);
    cout << "\nRectangle Name: " << shapeName << endl;
    cout << "X-coordinate: " << setw(9) << origin.getx() << endl;
    cout << "Y-coordinate: " << setw(9) << origin.gety() << endl;
    cout << "Side a: " << setw(15) << get_side_a() << endl;
    cout << "Side b: " << setw(15) << get_side_b() << endl;
    cout << "Area: " << setw(17) << area() << endl;
    cout << "Perimeter: "<< setw(12) << perimeter()  << endl;

}

Rectangle::Rectangle(const Rectangle& source):
    Square(source), Shape(source)
{
    side_b = source.get_side_b();
}

Rectangle& Rectangle::operator =(Rectangle&rhs){
    if(this != &rhs){
        Square::operator=(rhs);
        side_b = rhs.get_side_b();
    }
    return *this;
}
```

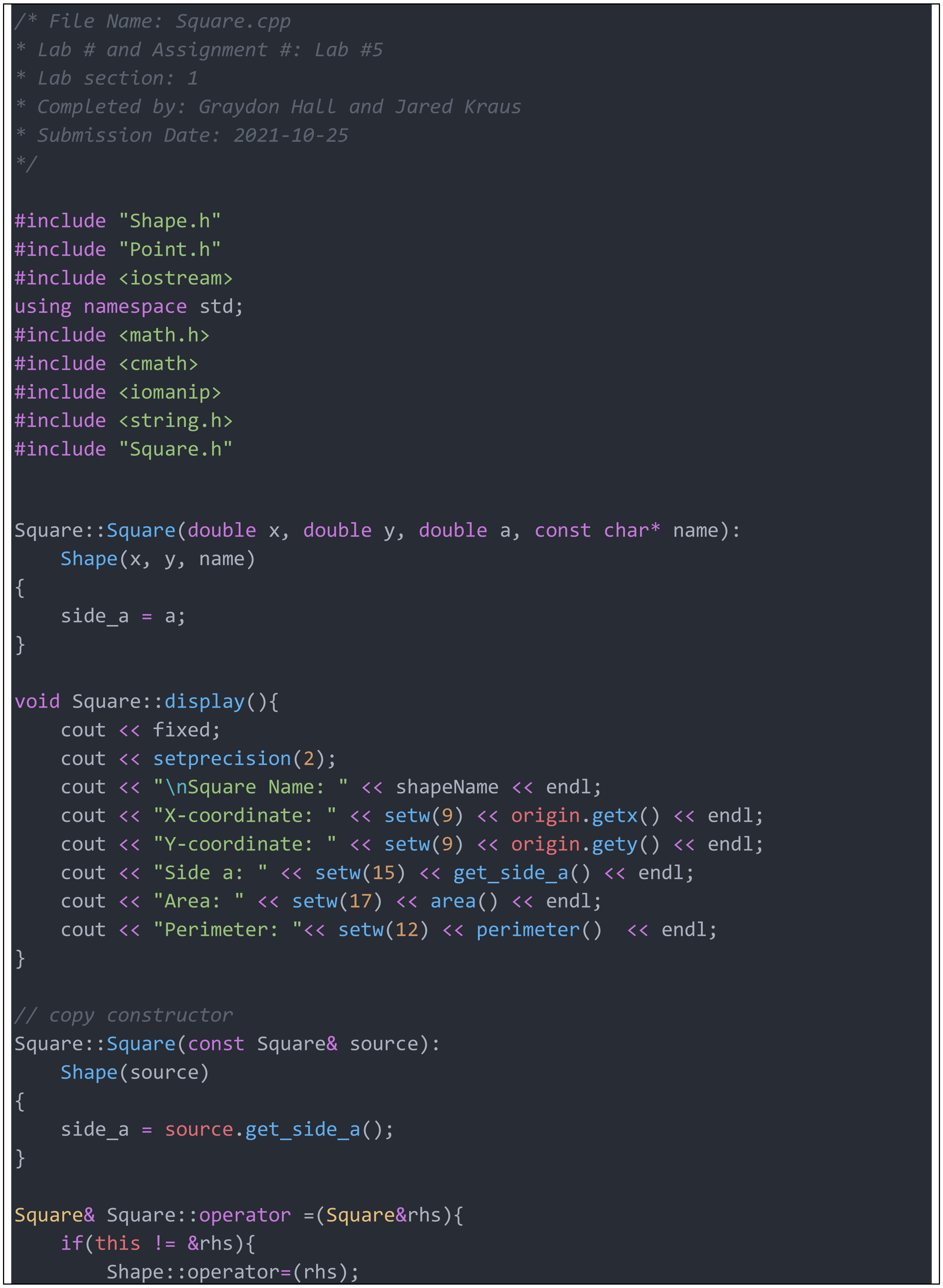
```
/* File Name: Rectangle.h
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#include "Point.h"
#include "Shape.h"
#include "Square.h"
```

```
#ifndef RECTANGLE
#define RECTANGLE
class Rectangle: public Square{

protected:
    double side_b;

public:
    Rectangle(double x, double y, double a, double b, const char* name);
    void display();
    double get_side_b() const{return side_b;}
    double area(){return side_a * side_b;}
    double perimeter(){return 2 * side_a + 2*side_b;}
    void set_side_b(double value){side_b = value;}
    Rectangle(const Rectangle& source);
    Rectangle& operator =(Rectangle&rhs);
};


#endif
```

```
/* File Name: Shape.cpp
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/
using namespace std;
#include <iostream>
#include <math.h>
#include <cmath>
#include <iomanip>
#include <string.h>
#include "Shape.h"
#include "Point.h"



Shape::Shape(double x, double y, const char* name):origin(x,y){
    int len = strlen(name);
    shapeName = new char[len];
    strcpy(shapeName, name);

}
```

```cpp
double Shape::distance (Shape& other){
    return origin.distance(other.origin);
}

double Shape::distance (Shape& the_shape, Shape& other){
    return Point::distance(the_shape.origin, other.origin);
}

void Shape::move (double dx, double dy){
    origin.setx(origin.getx()+dx);
    origin.sety(origin.gety()+dy);
}

// copy constructor
Shape::Shape(const Shape& source):
    origin(source.origin.getx(), source.origin.gety())
{
    int len = strlen(source.getName());
    shapeName = new char[len];
    strcpy(shapeName, source.getName());
}

// overload assignment operator
Shape& Shape::operator =(Shape&s){
    if(this!=&s){
        delete [] shapeName;
        origin.setx(s.origin.getx());
        origin.sety(s.origin.gety());
        int len = strlen(s.getName());
        shapeName = new char[len];
        strcpy(shapeName, s.getName());
    }
    return *this;
}

void Shape::display(){
    cout << fixed;
    cout << setprecision(2);
    cout << "\nShape Name: " << shapeName << endl;
    cout << "X-coordinate: " << setw(9) << origin.getx() << endl;
    cout << "Y-coordinate: " << setw(9) << origin.gety() << endl;
}
```

```cpp
/* File Name: Shape.h
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#include "Point.h"

#ifndef SHAPE
#define SHAPE
class Shape{

protected:
    Point origin;
    char * shapeName;

public:
    Shape(double x, double y, const char* name);
    ~Shape(){delete shapeName;}
    Shape(const Shape& source);
    Shape& operator =(Shape&s);

    double distance (Shape& other);
    static double distance (Shape& the_shape, Shape& other);
    void move (double dx, double dy);
    virtual void display();

    const Point & getOrigin() const{return origin;}
    char * getName() const{return shapeName;}

    // pure virtual functions... why do we get error if these aren't virtual?
    virtual double perimeter()=0;
    virtual double area()=0;
};


#endif
```



```
/* File Name: Square.cpp
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#include "Shape.h"
#include "Point.h"
#include <iostream>
using namespace std;
#include <math.h>
#include <cmath>
#include <iomanip>
#include <string.h>
#include "Square.h"


Square::Square(double x, double y, double a, const char* name):
    Shape(x, y, name)
{
    side_a = a;
}

void Square::display(){
    cout << fixed;
    cout << setprecision(2);
    cout << "\nSquare Name: " << shapeName << endl;
    cout << "X-coordinate: " << setw(9) << origin.getx() << endl;
    cout << "Y-coordinate: " << setw(9) << origin.gety() << endl;
    cout << "Side a: " << setw(15) << get_side_a() << endl;
    cout << "Area: " << setw(17) << area() << endl;
    cout << "Perimeter: "<< setw(12) << perimeter()  << endl;
}

// copy constructor
Square::Square(const Square& source):
    Shape(source)
{
    side_a = source.get_side_a();
}

Square& Square::operator =(Square&rhs){
    if(this != &rhs){
        Shape::operator=(rhs);
        side_a = rhs.get_side_a();
    }
    return *this;
}
```

```
/* File Name: Square.h
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#include "Point.h"
#include "Shape.h"

#ifndef SQUARE
#define SQUARE
class Square: virtual public Shape{

protected:
    double side_a;

public:
    Square(double x, double y, double side_a, const char* name);
    Square(const Square& source);
    Square& operator =(Square&rhs);

    void display();
    double area(){return side_a * side_a;}
    double perimeter(){return 4 * side_a;}

    void set_side_a(double value){side_a = value;}
    double get_side_a() const {return side_a;}

};


#endif
```

Exercise C

Program Output (copied from terminal)

```
Expected to dispaly the distance between m and n is: 3
The distance between m and n is: 3
Expected second version of the distance function also print: 3
The distance between m and n is again: 3

Testing Functions in class Square:

Square Name: SQUARE - S
X-coordinate:      5.00
Y-coordinate:      7.00
Side a:           12.00
Area:            144.00
Perimeter:        48.00

Testing Functions in class Rectangle:
Rectangle Name: RECTANGLE A
X-coordinate:      5.00
Y-coordinate:      7.00
Side a:           12.00
Side b:           15.00
Area:            180.00
Perimeter:        54.00

Rectangle Name: RECTANGLE B
X-coordinate:     16.00
Y-coordinate:      7.00
Side a:            8.00
Side b:            9.00
Area:             72.00
Perimeter:        34.00

Distance between square a, and b is: 11.00

Rectangle Name: RECTANGLE A
X-coordinate:      5.00
Y-coordinate:      7.00
Side a:           12.00
Side b:           15.00
Area:            180.00
Perimeter:        54.00

Testing assignment operator in class Rectangle:

Rectangle Name: RECTANGLE rec2
```

```
X-coordinate:     3.00
Y-coordinate:     4.00
Side a:          11.00
Side b:           7.00
Area:            77.00
Perimeter:       36.00

Expected to display the following values for objec rec2:
Rectangle Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Side a: 12
Side b: 15
Area: 180
Perimeter: 54

If it doesn't there is a problem with your assignment operator.


Rectangle Name: RECTANGLE A
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:          12.00
Side b:          15.00
Area:           180.00
Perimeter:       54.00

Testing copy constructor in class Rectangle:

Rectangle Name: RECTANGLE A
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:         100.00
Side b:         200.00
Area:         20000.00
Perimeter:      600.00

Expected to display the following values for objec rec2:
Rectangle Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Side a: 100
Side b: 200
Area: 20000
Perimeter: 600

If it doesn't there is a problem with your assignment operator.
```

```
Rectangle Name: RECTANGLE A
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:         100.00
Side b:         200.00
Area:         20000.00
Perimeter:      600.00

Testing array of pointers and polymorphism:

Square Name: SQUARE - S
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:          12.00
Area:           144.00
Perimeter:       48.00

Rectangle Name: RECTANGLE B
X-coordinate:    16.00
Y-coordinate:     7.00
Side a:           8.00
Side b:           9.00
Area:            72.00
Perimeter:       34.00

Rectangle Name: RECTANGLE A
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:          12.00
Side b:          15.00
Area:           180.00
Perimeter:       54.00

Rectangle Name: RECTANGLE A
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:         100.00
Side b:         200.00
Area:         20000.00
Perimeter:      600.00

Testing Functions in class Circle:

Circle Name: CIRCLE C
X-coordinate:     3.00
Y-coordinate:     5.00
Radius:           9.00
```

```
Area:           254.47
Perimeter:       56.55
the area of CIRCLE C is: 254.47
the perimeter of CIRCLE C is: 56.55

The distance between rectangle a and circle c is: 2.83

Curve Cut Name: CurveCut rc
X-coordinate:     6.00
Y-coordinate:     5.00
Side a:          10.00
Side b:          12.00
Cut Radius:       9.00
the area of CurveCut rc is: 56.38
the perimeter of CurveCut rc is: 40.14

The distance between rc and c is: 3.00

Square Name: SQUARE - S
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:          12.00
Area:           144.00
Perimeter:       48.00

the area of SQUARE - Sis: 144.00
the perimeter of SQUARE - S is: 48.00
Rectangle Name: RECTANGLE A
X-coordinate:     5.00
Y-coordinate:     7.00
Side a:         400.00
Side b:         300.00
Area:        120000.00
Perimeter:     1400.00

the area of RECTANGLE Ais: 120000.00
the perimeter of SQUARE - S is: 1400.00
Circle Name: CIRCLE C
X-coordinate:     3.00
Y-coordinate:     5.00
Radius:           9.00
Area:           254.47
Perimeter:       56.55

the area of CIRCLE Cis: 254.47
the circumference of CIRCLE C is: 56.55
Curve Cut Name: CurveCut rc
X-coordinate:     6.00
```

```
Y-coordinate:     5.00
Side a:          10.00
Side b:          12.00
Cut Radius:       9.00

the area of CurveCut rcis: 56.38
the perimeter of CurveCut rc is: 40.14
Testing copy constructor in class CurveCut:

Curve Cut Name: CurveCut rc
X-coordinate:     6.00
Y-coordinate:     5.00
Side a:          10.00
Side b:          12.00
Cut Radius:       9.00

Testing assignment operator in class CurveCut:

Curve Cut Name: CurveCut cc2
X-coordinate:     2.00
Y-coordinate:     5.00
Side a:         100.00
Side b:          12.00
Cut Radius:       9.00

Curve Cut Name: CurveCut rc
X-coordinate:     6.00
Y-coordinate:     5.00
Side a:          10.00
Side b:          12.00
Cut Radius:       9.00
```

Code files:

```cpp
/* File Name: Circle.cpp
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

using namespace std;
#include <iostream>
#include <math.h>
#include <cmath>
#include <iomanip>

#include "Shape.h"
#include "Point.h"
#include <string.h>
#include "Circle.h"


Circle::Circle(double x, double y, double r, const char* name):
    Shape(x, y, name)
{
    radius = r;
}

void Circle::display(){
    cout << fixed;
    cout << setprecision(2);
    cout << "\nCircle Name: " << shapeName << endl;
    cout << "X-coordinate: " << setw(9) << origin.getx() << endl;
    cout << "Y-coordinate: " << setw(9) << origin.gety() << endl;
    cout << "Radius: " << setw(15) << get_radius() << endl;
    cout << "Area: " << setw(17) << area() << endl;
    cout << "Perimeter: "<< setw(12) << perimeter()  << endl;

}
Circle::Circle(const Circle& source):
    Shape(source)
{
    radius = source.get_radius();
}

Circle& Circle::operator =(Circle &rhs){
    if(this != &rhs){
        Shape::operator=(rhs);
        radius = rhs.get_radius();
    }
    return *this;
}
```

```
/* File Name: Circle.h
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#include "Point.h"
#include "Shape.h"

#ifndef CIRCLE
#define CIRCLE

class Circle: virtual public Shape{

protected:
    double radius;

public:
    Circle(double x, double y, double r, const char* name);
    void display();
    double get_radius() const {return radius;}
    double area(){return 3.14159265 * radius * radius;}
    double perimeter(){return 3.14159265 * 2 * radius;}
    void set_radius(double value){radius = value;}
    Circle(const Circle& source);
    Circle& operator =(Circle&rhs);
};


#endif
```

```
/* File Name: CurveCut.cpp
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

using namespace std;

#include <iostream>
#include <math.h>
#include <cmath>
#include <iomanip>
#include <string.h>
#include <assert.h>

#include "Point.h"
#include "Shape.h"
#include "Square.h"
#include "Rectangle.h"
#include "CurveCut.h"


CurveCut::CurveCut(double x, double y, double a, double b, double r, const
char* name):
    Rectangle(x, y, a, b, name), Circle(x, y, r, name), Shape(x, y, name){
        if(r>a || r>b){
            cerr << "Error: Radius cannot be bigger than either of rectangle
sides" << endl;
            exit(1);
        }
    }

double CurveCut::area(){
    return Rectangle::area() - 0.25*Circle::area();
}

double CurveCut::perimeter(){
    return Rectangle::perimeter() +  0.25*Circle::perimeter() - 2*radius;
}

// copy constructor
CurveCut::CurveCut(const CurveCut& source):
    Shape(source), Rectangle(source), Circle(source){
```

```cpp
}

//overload equals sign
CurveCut& CurveCut::operator =(CurveCut&rhs){
    if(this != &rhs){
        Rectangle::operator=(rhs);
        Circle::operator=(rhs);
    }
    return *this;
}

void CurveCut::display(){
    cout << fixed;
    cout << setprecision(2);
    cout << "\nCurve Cut Name: " << shapeName << endl;
    cout << "X-coordinate: " << setw(9) << origin.getx() << endl;
    cout << "Y-coordinate: " << setw(9) << origin.gety() << endl;
    cout << "Side a: " << setw(15) << get_side_a() << endl;
    cout << "Side b: " << setw(15) << get_side_b() << endl;
    cout << "Cut Radius: "<< setw(11) << get_radius()  << endl;
}
```

```cpp
/* File Name: CurveCut.h
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/
#include "Point.h"
#include "Rectangle.h"
#include "Circle.h"

#ifndef CURVECUT
#define CURVECUT

class CurveCut: public Rectangle, public Circle{

protected:

public:
    CurveCut(double x, double y, double side_a, double side_b, double r, const
char* name);
    void display();
    double area();
    double perimeter();
    CurveCut(const CurveCut& source);
    CurveCut& operator =(CurveCut&s);
};


#endif
```

```
/* File Name: GraphicsWorld.cpp
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#include "GraphicsWorld.h"
#include "Point.h"
#include "Shape.h"
#include "Square.h"
#include <iostream>
using namespace std;
#include "Rectangle.h"
#include "Circle.h"
#include "CurveCut.h"

void GraphicsWorld::run(){
    #if 1 // Change 0 to 1 to test Point
        Point m (6, 8);
        Point n (6,8);
        n.setx(9);
        cout << "\nExpected to dispaly the distance between m and n is: 3";
        cout << "\nThe distance between m and n is: " << m.distance(n);
        cout << "\nExpected second version of the distance function also print:
3";
        cout << "\nThe distance between m and n is again: "
        << Point::distance(m, n);
    #endif // end of block to test Point
    #if 1 // Change 0 to 1 to test Square
        cout << "\n\nTesting Functions in class Square:" <<endl;
        Square s(5, 7, 12, "SQUARE - S");
        s.display();
    #endif // end of block to test Square
    #if 1 // Change 0 to 1 to test Rectangle
        cout << "\nTesting Functions in class Rectangle:";
        Rectangle a(5, 7, 12, 15, "RECTANGLE A");
        a.display();
        Rectangle b(16 , 7, 8, 9, "RECTANGLE B");
        b.display();
        double d = a.distance(b);
        cout <<"\nDistance between square a, and b is: " << d << endl;
        Rectangle rec1 = a;
        rec1.display();
        cout << "\nTesting assignment operator in class Rectangle:" <<endl;
        Rectangle rec2 (3, 4, 11, 7, "RECTANGLE rec2");
        rec2.display();
        rec2 = a;
        a.set_side_b(200);
        a.set_side_a(100);
        cout << "\nExpected to display the following values for objec rec2: "
<< endl;
        cout << "Rectangle Name: RECTANGLE A\n" << "X-coordinate: 5\n" << "Y-
coordinate: 7\n"
        << "Side a: 12\n" << "Side b: 15\n" << "Area: 180\n" << "Perimeter:
54\n" ;
        cout << "\nIf it doesn't there is a problem with your assignment
operator.\n" << endl;
        rec2.display();
        cout << "\nTesting copy constructor in class Rectangle:" <<endl;
        Rectangle rec3 (a);
        rec3.display();
        a.set_side_b(300);
        a.set_side_a(400);
        cout << "\nExpected to display the following values for objec rec2: "
<< endl;
        cout << "Rectangle Name: RECTANGLE A\n" << "X-coordinate: 5\n" << "Y-
coordinate: 7\n"
        << "Side a: 100\n" << "Side b: 200\n" << "Area: 20000\n" << "Perimeter:
600\n" ;
        cout << "\nIf it doesn't there is a problem with your assignment
operator.\n" << endl;
        rec3.display();
    #endif // end of block to test Rectangle
    #if 1 // Change 0 to 1 to test using array of pointer and polymorphism
        cout << "\nTesting array of pointers and polymorphism:" <<endl;
        Shape* sh1[4];
        sh1[0] = &s;
        sh1[1] = &b;
        sh1 [2] = &rec1;
        sh1 [3] = &rec3;
        sh1 [0]->display();
        sh1 [1]->display();
        sh1 [2]->display();
        sh1 [3]->display();
    #endif // end of block to test array of pointer and polymorphism


    #if 1
        cout << "\nTesting Functions in class Circle:" <<endl;
        Circle c (3, 5, 9, "CIRCLE C");
        c.display();
        cout << "the area of " << c.getName() <<" is: "<< c.area() << endl;
        cout << "the perimeter of " << c.getName() << " is: "<< c.perimeter()
<< endl;
        d = a.distance(c);
        cout << "\nThe distance between rectangle a and circle c is: " <<d<<
endl;
        CurveCut rc (6, 5, 10, 12, 9, "CurveCut rc");
        rc.display();
        cout << "the area of " << rc.getName() <<" is: "<< rc.area()<< endl;
        cout << "the perimeter of " << rc.getName() << " is: "<<
rc.perimeter()<< endl;
        d = rc.distance(c);
        cout << "\nThe distance between rc and c is: " <<d<< endl;
        // Using array of Shape pointers:
        Shape* sh[4];
        sh[0] = &s;
        sh[1] = &a;
        sh [2] = &c;
        sh [3] = &rc;
        sh [0]->display();
        cout << "\nthe area of "<< sh[0]->getName() << "is: "<< sh[0] ->area();
        cout << "\nthe perimeter of " << sh[0]->getName () << " is: "<< sh[0]-
>perimeter();
        sh [1]->display();
        cout << "\nthe area of "<< sh[1]->getName() << "is: "<< sh[1] ->area();
        cout << "\nthe perimeter of " << sh[0]->getName () << " is: "<< sh[1]-
>perimeter();
        sh [2]->display();
        cout << "\nthe area of "<< sh[2]->getName() << "is: "<< sh[2] ->area();
        cout << "\nthe circumference of " << sh[2]->getName ()<< " is: "<<
sh[2]->perimeter();
        sh [3]->display();
        cout << "\nthe area of "<< sh[3]->getName() << "is: "<< sh[3] ->area();
        cout << "\nthe perimeter of " << sh[3]->getName () << " is: "<< sh[3]-
>perimeter();
```

```
        cout << "\nTesting copy constructor in class CurveCut:" <<endl;
        CurveCut cc = rc;
        cc.display();
        cout << "\nTesting assignment operator in class CurveCut:" <<endl;
        CurveCut cc2(2, 5, 100, 12, 9, "CurveCut cc2");
        cc2.display();
        cc2 = cc;
        cc2.display();
    #endif

}

int main(){
    GraphicsWorld x;
    x.run();
    return 0;
}
```

```
/* File Name: GraphicsWorld.h
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#ifndef GRAPHICS_WORLD
#define GRAPHICS_WORLD

class GraphicsWorld{
public:
    void run();

};


#endif
```

```
/* File Name: Point.cpp
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#include "Point.h"
```

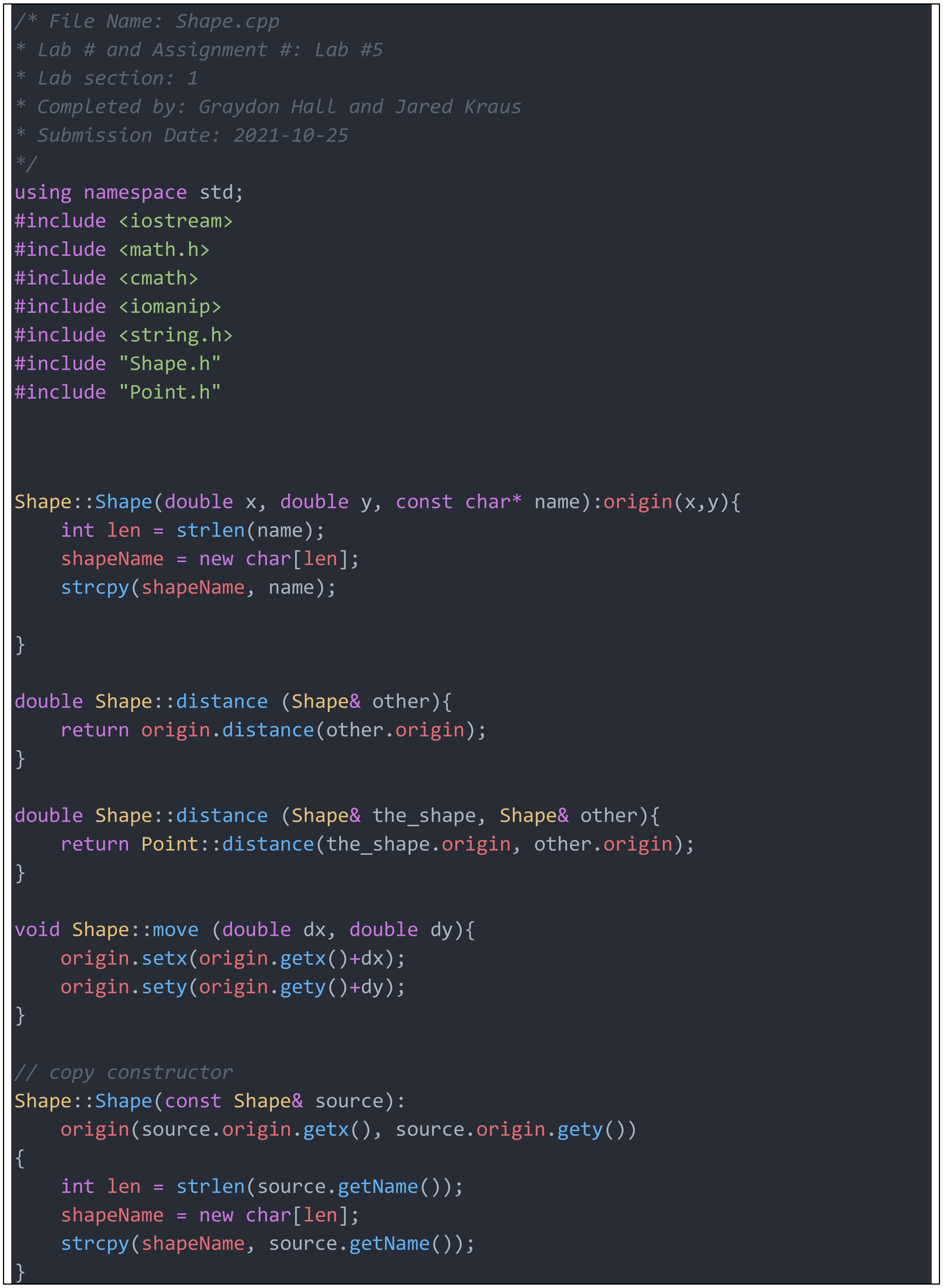
```cpp
#include <iostream>
using namespace std;
#include <math.h>
#include <cmath>
#include <iomanip>

int Point::point_counter=0;
int Point::id_counter=1000;

Point::Point(double x, double y){
    xcoordinate = x;
    ycoordinate = y;
    point_counter++;
    id_counter++;
    pointID = id_counter;
}

void Point::display(){
    cout << fixed;
    cout << setprecision(2);
    cout << "\nX-coordinate: " << setw(9) << xcoordinate << endl;
    cout << "Y-coordinate: " << setw(9) << ycoordinate << endl;
}

double Point::distance(const Point& m, const Point& n){
    // Pass m and n by reference to uneccesary destructor call for them
    return sqrt(pow(abs(m.getx() - n.getx()),2)+pow(abs(m.gety() -
n.gety()),2));
}

double Point::distance(const Point &p){
    // Pass p by reference to uneccesary destructor call for it
    return sqrt(pow(abs(getx() - p.getx()),2)+pow(abs(gety() - p.gety()),2));
}

Point::~Point(){
    point_counter --;
}
```

```
/* File Name: Point.h
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#ifndef POINT
#define POINT

class Point{
private:
    double xcoordinate;
    double ycoordinate;
    int pointID;
    static int point_counter;  // counter for # of points created
    static int id_counter; // assign IDs to each point
public:
    Point(double x, double y);
    ~Point();
    static double distance(const Point& m, const Point& n);
    double distance(const Point &p);
    static int counter(){return point_counter;}
    void display();

    void setx(double value){xcoordinate=value;}
    void sety(double value){ycoordinate=value;}
    double getx() const{return xcoordinate;}
    double gety() const{return ycoordinate;}
    int getID() const{return pointID;}

};


#endif
```

```cpp
/* File Name: Rectangle.cpp
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

using namespace std;
#include <iostream>
#include <math.h>
#include <cmath>
#include <iomanip>
#include <string.h>

#include "Square.h"
#include "Rectangle.h"
#include "Shape.h"
#include "Point.h"


Rectangle::Rectangle(double x, double y, double a, double b, const char* name):
    Square(x, y, a, name), Shape(x, y, name)
{
    side_b = b;
}

void Rectangle::display(){
    cout << fixed;
    cout << setprecision(2);
    cout << "\nRectangle Name: " << shapeName << endl;
    cout << "X-coordinate: " << setw(9) << origin.getx() << endl;
    cout << "Y-coordinate: " << setw(9) << origin.gety() << endl;
    cout << "Side a: " << setw(15) << get_side_a() << endl;
    cout << "Side b: " << setw(15) << get_side_b() << endl;
    cout << "Area: " << setw(17) << area() << endl;
    cout << "Perimeter: "<< setw(12) << perimeter()  << endl;

}

Rectangle::Rectangle(const Rectangle& source):
    Square(source), Shape(source)
{
    side_b = source.get_side_b();
}
```

```cpp
Rectangle& Rectangle::operator =(Rectangle&rhs){
    if(this != &rhs){
        Square::operator=(rhs);
        side_b = rhs.get_side_b();
    }
    return *this;
}
```

```cpp
/* File Name: Rectangle.h
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#include "Point.h"
#include "Shape.h"
#include "Square.h"

#ifndef RECTANGLE
#define RECTANGLE
class Rectangle: public Square{

protected:
    double side_b;

public:
    Rectangle(double x, double y, double a, double b, const char* name);
    void display();
    double get_side_b() const{return side_b;}
    double area(){return side_a * side_b;}
    double perimeter(){return 2 * side_a + 2*side_b;}
    void set_side_b(double value){side_b = value;}
    Rectangle(const Rectangle& source);
    Rectangle& operator =(Rectangle&rhs);
};


#endif
```



```cpp
/* File Name: Shape.cpp
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/
using namespace std;
#include <iostream>
#include <math.h>
#include <cmath>
#include <iomanip>
#include <string.h>
#include "Shape.h"
#include "Point.h"



Shape::Shape(double x, double y, const char* name):origin(x,y){
    int len = strlen(name);
    shapeName = new char[len];
    strcpy(shapeName, name);

}

double Shape::distance (Shape& other){
    return origin.distance(other.origin);
}

double Shape::distance (Shape& the_shape, Shape& other){
    return Point::distance(the_shape.origin, other.origin);
}

void Shape::move (double dx, double dy){
    origin.setx(origin.getx()+dx);
    origin.sety(origin.gety()+dy);
}

// copy constructor
Shape::Shape(const Shape& source):
    origin(source.origin.getx(), source.origin.gety())
{
    int len = strlen(source.getName());
    shapeName = new char[len];
    strcpy(shapeName, source.getName());
}
```

```
// overload assignment operator
Shape& Shape::operator =(Shape&s){
    if(this!=&s){
        delete [] shapeName;
        origin.setx(s.origin.getx());
        origin.sety(s.origin.gety());
        int len = strlen(s.getName());
        shapeName = new char[len];
        strcpy(shapeName, s.getName());
    }
    return *this;
}

void Shape::display(){
    cout << fixed;
    cout << setprecision(2);
    cout << "\nShape Name: " << shapeName << endl;
    cout << "X-coordinate: " << setw(9) << origin.getx() << endl;
    cout << "Y-coordinate: " << setw(9) << origin.gety() << endl;
}
```

```
/* File Name: Shape.h
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#include "Point.h"

#ifndef SHAPE
#define SHAPE
class Shape{

protected:
    Point origin;
    char * shapeName;

public:
    Shape(double x, double y, const char* name);
    ~Shape(){delete shapeName;}
    Shape(const Shape& source);
```

```cpp
    Shape& operator =(Shape&s);

    double distance (Shape& other);
    static double distance (Shape& the_shape, Shape& other);
    void move (double dx, double dy);
    virtual void display();

    const Point & getOrigin() const{return origin;}
    char * getName() const{return shapeName;}

    // pure virtual functions... why do we get error if these aren't virtual?
    virtual double perimeter()=0;
    virtual double area()=0;
};


#endif
```

```cpp
/* File Name: Square.cpp
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#include "Shape.h"
#include "Point.h"
#include <iostream>
using namespace std;
#include <math.h>
#include <cmath>
#include <iomanip>
#include <string.h>
#include "Square.h"


Square::Square(double x, double y, double a, const char* name):
    Shape(x, y, name)
{
    side_a = a;
}

void Square::display(){
    cout << fixed;
    cout << setprecision(2);
    cout << "\nSquare Name: " << shapeName << endl;
    cout << "X-coordinate: " << setw(9) << origin.getx() << endl;
    cout << "Y-coordinate: " << setw(9) << origin.gety() << endl;
    cout << "Side a: " << setw(15) << get_side_a() << endl;
    cout << "Area: " << setw(17) << area() << endl;
    cout << "Perimeter: "<< setw(12) << perimeter()  << endl;
}

// copy constructor
Square::Square(const Square& source):
    Shape(source)
{
    side_a = source.get_side_a();
}

Square& Square::operator =(Square&rhs){
    if(this != &rhs){
        Shape::operator=(rhs);
        side_a = rhs.get_side_a();
    }
    return *this;
}
```

```
/* File Name: Square.h
* Lab # and Assignment #: Lab #5
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-10-25
*/

#include "Point.h"
#include "Shape.h"

#ifndef SQUARE
#define SQUARE
class Square: virtual public Shape{

protected:
    double side_a;

public:
    Square(double x, double y, double side_a, const char* name);
    Square(const Square& source);
    Square& operator =(Square&rhs);

    void display();
    double area(){return side_a * side_a;}
    double perimeter(){return 4 * side_a;}

    void set_side_a(double value){side_a = value;}
    double get_side_a() const {return side_a;}

};


#endif
```