Lab 5 Report

Course: ENSF 619 - Fall 2020
Lab #: Lab 5
Student Name: Davis Allan, 10016543
Submission Date: Oct 23 2020

Exercise A:

Source Codes:

Class Point

```
 /*
* File Name: point.h
* Lab # and Assignment #: Lab 5 Exercise A
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/

#ifndef POINT_H
#define POINT_H

class Point {
    public:
        Point(double x, double y);
        // PROMISES: Point object created with all x and y initialized
from
        //the given arguments. Increments the counter for the number of
points
        //existence

        ~Point();
        // PROMISES: decrement the counter for the number of points in
existence

        Point(const Point& source);
        //  REQUIRES: source is a reference to a Point object
        //  PROMISES: constructs a new Point object and deep copies the
data members
        //  of the source to the newly constructed Shape object.
Increments the point counter

        Point& operator=(const Point& rhs);
        //  REQUIRES: rhs is a reference to a Point object
        //  PROMISES: copy and assign the data members of rhs object to
the Point object invoking
        //  this assignment
```

```
        const double getX() const {return xCoord;};
        // PROMISES: returns the x-coordinate of the point

        void setX(double x) {xCoord = x;};
        // PROMISES: sets the point object's x-coordinate to given the
argument

        const double getY() const {return yCoord;};
        // PROMISES: returns the y-coordinate of the point

        void setY(double y) {yCoord = y;};
        // PROMISES: sets the Point object's x-coordinate to given the
argument

        const int getID() {return id;};
        // PROMISES: returns the unique id of the point

        void display() const;
        // PROMISES: displays the point objects data members to the
console

        static int getCounter();
        // PROMISES: returns the number of objects of class Point

        double dist(const Point& a) const;
        // PROMISES: calculates the distance between the object invoking
the method,
        // and the point in the argument

        static double dist(const Point& a, const Point& b);
        // PROMISES: calculates and returns the distance between the two
points

    private:
        static int counter;
        int id;
        double xCoord;
        double yCoord;
};

#endif
```

```cpp
/*
* File Name: point.cpp
* Lab # and Assignment #: Lab 5 Exercise A
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/

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

int Point::counter = 0;

Point::Point(double x, double y): xCoord(x), yCoord(y) {
    id = 1001 + counter++;
}

Point::~Point(){
    counter--;
}

Point::Point(const Point& source): xCoord(source.getX()),
yCoord(source.getY()) {
    id = 1001 + counter++;
}

Point& Point::operator=(const Point& rhs) {
    if (this != &rhs) {
        xCoord = rhs.getX();
        yCoord = rhs.getY();
        id = 1001 + counter++;
    }
    return *this;
}

void Point::display() const {
    streamsize def = cout.precision();
    cout.setf(ios::fixed);
    cout.precision(2);
    cout << "X-coordinate: " << getX() << endl;
```

```
    cout << "Y-coordinate: " << getY() << endl;
    cout.unsetf(ios::fixed);
    cout.precision(def);
}

int Point::getCounter() {
    return counter;
}

double Point::dist(const Point& a) const {
    return sqrt(pow((getX() - a.getX()), 2) + (pow((getY() -
a.getY()),2)));
}

double Point::dist(const Point& a, const Point& b) {
    return sqrt(pow((a.getX() - b.getX()), 2) + (pow((a.getY()-
b.getY()),2)));
}
```

Class Shape

```
/*
* File Name: shape.h
* Lab # and Assignment #: Lab 5 Exercise A
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/

#include "point.h"

#ifndef SHAPE_H
#define SHAPE_H

class Shape {
    public:
        Shape(double x, double y, const char *shapeName);
        //  REQUIRES: shapeName points to first char of built-in string
        //  PROMISES: Shape object created with its data members initialized from
        //  the provided arguments

        virtual ~Shape();
        //  PROMISES: deallocates the dynamically allocated char array for the shape name

        Shape(const Shape& source);
        //  REQUIRES: source is a reference to a Shape object
        //  PROMISES: constructs a new Shape object and deep copies the data members
        //  of the source to the newly constructed Shape object

        Shape& operator=(const Shape& rhs);
        //  REQUIRES: rhs is a reference to a Shape object
        //  PROMISES: copy and assign the data members of rhs object to the Shape object invoking
        //  this assignment

        const Point& getOrigin() const {return origin;};
        //  PROMISES: returns a read only reference to the Point object representing the shapes origin
```

```
        const char* getName() const {return shapeName;};
        //  PROMISES: returns a read only pointer to the built in string
containing the name of the shape

        virtual void display() const;
        //  PROMISES: displays the Shapes

        virtual double area() const = 0;
        //  PROMISES: returns the area of the Shape

        virtual double perimeter() const = 0;
        //  PROMISES: returns the perimeter of the Shape

        double distance (const Shape& other) const;
        // REQUIRES: other is a reference to a Shape object
        // PROMISES: returns the calculated distance between the object
invoking this method and the
        // other shape

        static double distance(const Shape& the_shape, const Shape&
other);
        //  REQUIRES: the_shape and other are references to Shape objects
        //  PROMISES: returns the calculated distance between the
the_shape and other Shape objects

        void move(double dx, double dy);
        // PROMISES: updates the Shape object's origin by dx and dy

    protected:
        Point origin;
        char *shapeName;
};

#endif
```

```
/*
* File Name: shape.cpp
* Lab # and Assignment #: Lab 5 Exercise A
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/
```

```cpp
#include "point.h"
#include "shape.h"
#include <iostream>
#include <math.h>
#include <string.h>
#include <assert.h>
using namespace std;

Shape::Shape(double x, double y, const char *name): origin(Point(x,y)){
    shapeName = new char[strlen(name) + 1];
    assert(shapeName != NULL);
    strcpy(shapeName, name);
}

Shape::~Shape() {
    delete [] shapeName;
    shapeName = NULL;
}

Shape::Shape(const Shape& source): origin(Point(source.getOrigin().getX(),
source.getOrigin().getY())) {
    shapeName = new char[strlen(source.getName())];
    assert(shapeName != NULL);
    strcpy(shapeName, source.getName());
}

Shape& Shape::operator=(const Shape& rhs) {
    if (this != &rhs) {
        delete [] shapeName;
        shapeName = new char [strlen(rhs.getName())];
        assert(shapeName != NULL);
        strcpy(shapeName, rhs.getName());
        origin = Point(rhs.getOrigin().getX(), rhs.getOrigin().getY());
    }
    return *this;
}

void Shape::display() const {
    cout << "Shape Name: " << getName() << endl;
    getOrigin().display();
}
```

```
double Shape::distance(const Shape& other) const {
    return getOrigin().dist(other.getOrigin());
}

double Shape::distance(const Shape& the_shape, const Shape& other) {
    return the_shape.getOrigin().dist(the_shape.getOrigin(),
other.getOrigin());
}

void Shape::move(double dx, double dy) {
    origin.setX(getOrigin().getX() + dx);
    origin.setY(getOrigin().getY() + dy);
}
```

Class Square

```
/*
* File Name: square.h
* Lab # and Assignment #: Lab 5 Exercise A
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/

#include "shape.h"

#ifndef SQUARE_H
#define SQUARE_H

class Square: virtual public Shape {
    public:
        Square(double x, double y, double side, const char *name);
        //  REQUIRES: name is pointing to the first char of a built-in
string
        //  PROMISES: constructs a Square object with its data member
initialized
        //  from the provided arguments

        const double getSideA() const {return sideA;};
        //  PROMISES: returns the length of sideA

        void setSideA(double newSide) {sideA = newSide;};
        //  PROMISES: sets the length of sideA to the provided argument

        double area() const;
        //  PROMISES: returns the area of the Rectangle

        double perimeter() const;
        //  PROMISES: returns the perimeter of the Rectangle

        void display() const;
        //  PROMISES: display all data members to the console
    protected:
        double sideA;
};

#endif
```

```cpp
/*
* File Name: square.cpp
* Lab # and Assignment #: Lab 5 Exercise A
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/

#include "square.h"
#include <iostream>
using namespace std;

Square::Square(double x, double y, double side, const char *name)
       : Shape(x, y, name), sideA(side) {}

double Square::area() const{
    return getSideA() * getSideA();
}

double Square::perimeter() const {
    return getSideA() * 4;
}

void Square::display() const {
    cout << "Square Name: " << getName() << endl;
    getOrigin().display();
    cout << "Side a: " << getSideA() << endl;
    cout << "Area: " << area() << endl;
    cout << "Perimeter: " << perimeter() << endl;
}
```

Class Rectangle

```
/*
* File Name: rectangle.h
* Lab # and Assignment #: Lab 5 Exercise A
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/

#include "square.h"

#ifndef RECTANGLE_H
#define RECTANGLE_H

class Rectangle: public Square {
    public:
        Rectangle(double x, double y, double sideA, double sideB, const
char *name);
        //  REQUIRES: name is pointing to the first char of a built-in
string
        //  PROMISES: constructs a Rectangle object with its data member
initialized
        //  from the provided arguments

        const double getSideB() const {return sideB;};
        //  PROMISES: returns the length of sideB

        void setSideB(double newSideB) {sideB = newSideB;};
        //  PROMISES: sets the length of sideB to the provided argument

        double area() const;
        //  PROMISES: returns the area of the Rectangle

        double perimeter() const;
        //  PROMISES: returns the perimeter of the Rectangle

        void display() const;
        //  PROMISES: display all data members to the console

    protected:
        double sideB;
};
```

```
#endif
```

```cpp
/*
* File Name: rectangle.cpp
* Lab # and Assignment #: Lab 5 Exercise A
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/

#include "rectangle.h"
#include "square.h"
#include <math.h>
#include <iostream>
using namespace std;

Rectangle::Rectangle(double x, double y, double sideA, double sideB, const
char *name)
        : Shape(x, y, name), Square(x, y, sideA, name), sideB(sideB) {
}

double Rectangle::area() const {
    return getSideA() * getSideB();
}

double Rectangle::perimeter() const {
    return (2 * getSideA()) + (2 * getSideB());
}

void Rectangle::display() const {
    cout << "Rectangle Name: " << getName() << endl;
    getOrigin().display();
    cout << "Side a: " << getSideA() << endl;
    cout << "Side b: " << getSideB() << endl;
    cout << "Area: " << area() << endl;
    cout << "Perimiter: " << perimeter() << endl;
}
```

Class GraphicsWorld

```
/*
* File Name: graphicsWorld.h
* Lab # and Assignment #: Lab 5 Exercise A
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/

#ifndef GRAPHICS_WORLD_H
#define GRAPHICS_WORLD_H

class GraphicsWorld {
    public:
        void run();
};

#endif
```

```
/*
* File Name: graphicsWorld.cpp
* Lab # and Assignment #: Lab 5 Exercise A/B
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/

#include "point.h"
#include "graphicsWorld.h"
#include "square.h"
#include "rectangle.h"
#include "circle.h"
#include "curveCut.h"
#include <iostream>
using namespace std;

void GraphicsWorld::run() {
    cout << "\nProgram created by Davis Allan, 10016543\n\n";

#if 1 // Change 0 to 1 to test Point

    cout << "Testing functions in Class Point" << endl;
```

```
    Point m (6,8);
    Point n (6,8);
    cout << "\nTesting getCounter(), expected to display 2";
    cout << "\nThe number of points currently in existence are: " <<
Point::getCounter();
    cout << "\n\nExpected to display the point m, and expected output
is:";
    cout << "\nX-coordinate: 6.00";
    cout << "\nY-coordinate: 8.00\n\n";

    m.display();
    n.setX(9);

    cout << "\nExpected to display the distance between m and n is: 3";
    cout << "\nThe distance between m and n is: " << m.dist(n);
    cout << "\nExpected second version of the distance function also
print: 3";
    cout << "\nThe distance between m and n is again: " << Point::dist(m,
n);

    Point o (10,12);
    Point p (5,10);
    cout << "\n\nTesting getCounter(), expected to display 4";
    cout << "\nThe number of points currently in existence are: " <<
Point::getCounter();
    cout << "\n\nDisplaying most recent point's id, expecting 1004";
    cout << "\nUnique id of Point p is: " << p.getID();
#endif // end of block to test Point

#if 1 // Change 0 to 1 to test Square

    cout << "\n-----------------------------------------------------------";
    cout << "\nTesting Functions in class Square:\n\n";

    Square s(5, 7, 12, "SQUARE - S");
    s.display();
    cout << "\nTesting the move function, moving square a dx = 5, dy = -4"
        << "\nExpecting square to now be at coordinates x = 10.00, y =
3.00:\n\n";
    s.move(5, -4);
    s.display();
```

```
    cout << "\n\nTesting the copy constructor\n" << endl;
    Square sq1(s);
    s.setSideA(20);

    cout << "Expecting the following result for displaying sq1:\n"
         << "Square Name: SQUARE - S\n" << "X-coordinate: 10.00\n"
         << "Y-coordinate: 3.00\n" << "Side a: 12\n" << "Area: 144\n"
         << "Perimeter: 48\n\n";

    sq1.display();

    cout << "\nTesting the assignment operator:\n" << endl;

    Square sq2(1,2,15,"sq2");
    sq2 = s;
    s.setSideA(10);

    cout << "Expecting the following result for displaying sq2:\n"
         << "Square Name: SQUARE - S\n" << "X-coordinate: 10.00\n"
         << "Y-coordinate: 3.00\n" << "Side a: 20\n" << "Area: 400\n"
         << "Perimeter: 80\n\n";

    sq2.display();
    cout << "\n-------------------------------------------------------------";
#endif // end of block to test Square

#if 1 // Change 0 to 1 to test Rectangle

    cout << "\n\nTesting Functions in class Rectangle:\n\n";

    Rectangle a(5, 7, 12, 15, "RECTANGLE A");
    a.display();
    cout << endl;
    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 << endl;

    Rectangle rec1 = a;
    rec1.display();

    cout << "\nTesting assignment operator in class Rectangle:\n" <<endl;
```

```
    Rectangle rec2 (3, 4, 11, 7, "RECTANGLE rec2");
    rec2.display();
    rec2 = a;
    a.setSideB(200);
    a.setSideA(100);

    cout << "\nExpected to display the following values for objec rec2: "
<< endl;
    cout << "Rectangle Name: RECTANGLE A\n" << "X-coordinate: 5.00\n" <<
"Y-coordinate: 7.00\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.setSideB(300);
    a.setSideA(400);

    cout << "\nExpected to display the following values for objec rec3: "
<< endl;
    cout << "Rectangle Name: RECTANGLE A\n" << "X-coordinate: 5.00\n" <<
"Y-coordinate: 7.00\n"
    << "Side a: 100\n" << "Side b: 200\n" << "Area: 20000\n" <<
"Perimeter: 600\n" ;

    rec3.display();
    cout << "\n-----------------------------------------------------------";
#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* sh[4];
    sh[0] = &s;
    sh[1] = &b;
    sh [2] = &rec1;
    sh [3] = &rec3;
```

```
    sh [0]->display();
    sh [1]->display();
    sh [2]->display();
    sh [3]->display();
    cout << "\n------------------------------------------------------------";
#endif // end of block to test array of pointer and polymorphism

//EXERCISE B CODE:

#if 1
    cout << "\nTesting Functions in class Circle:" <<endl;

    Circle c (3, 5, 9, "CIRCLE C");
    c.display();

    cout << "\nthe 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 << endl;
    cout << "\n------------------------------------------------------------";
    cout << "\n\nTesting Functions in class CurveCut:\n" <<endl;

    CurveCut rc (6, 5, 10, 12, 9, "CurveCut rc");
    rc.display();



    cout << "the area of " << rc.getName() <<" is: "<< rc.area();
    cout << "\nthe perimeter of " << rc.getName() << " is: "<<
rc.perimeter();

    d = rc.distance(c);

    cout << "\n\nThe distance between rc and c is: " << d << endl;

    cout << "\nTesting copy constructor in class CurveCut:" <<endl;
    cout << "\nExpected to display the following values for objec cc: " <<
endl;
    cout << "Rectangle Name: CurveCurt rc\n" << "X-coordinate: 6.00\n" <<
```

```
"Y-coordinate: 5.00\n"
    << "Width: 12\n" << "Length: 10\n" << "Radius of the cut: 9\n" <<
endl;
    CurveCut cc (rc);
    rc.setSideB(40);
    cc.display();


    cout << "\nTesting assignment operator in class CurveCut:" <<endl;

    CurveCut cc2(2, 5, 100, 12, 9, "CurveCut cc2");

    cc2.display();
    cout << endl;
    cc2 = cc;
    cc2.display();
    cout << "\n-----------------------------------------------------------
\n";
    cout << "Testing array of pointers and polymorphism:" << endl;
    // Using array of Shape pointers:
    Shape* sha[4];
    sha[0] = &s;
    sha[1] = &a;
    sha [2] = &c;
    sha [3] = &rc;
    sha [0]->display();

    cout << "\nthe area of "<< sha[0]->getName() << " is: "<< sha[0] -
>area();
    cout << "\nthe perimeter of " << sha[0]->getName () << " is: "<<
sha[0]->perimeter() << endl << endl;

    sha [1]->display();

    cout << "\nthe area of "<< sha[1]->getName() << " is: "<< sha[1] -
>area();
    cout << "\nthe perimeter of " << sha[1]->getName () << " is: "<<
sha[1]->perimeter() << endl << endl;

    sha [2]->display();

    cout << "\nthe area of "<< sha[2]->getName() << " is: "<< sha[2] -
>area();
```

```
    cout << "\nthe circumference of " << sha[2]->getName ()<< " is: "<<
sha[2]->perimeter() << endl << endl;;

    sha [3]->display();

    cout << "\nthe area of "<< sha[3]->getName() << " is: "<< sha[3] -
>area() << endl;
    cout << "the perimeter of " << sha[3]->getName () << " is: "<< sha[3]-
>perimeter() << endl;
    cout << "\n-------------------------------------------------------
\n";



#endif

}

int main() {
    GraphicsWorld().run();
}
```

Program Output for Exercise A:

```
Program created by Davis Allan, 10016543

Testing functions in Class Point

Testing getCounter(), expected to display 2
The number of points currently in existence are: 2

Expected to display the point m, and expected output is:
X-coordinate: 6.00
Y-coordinate: 8.00

X-coordinate: 6.00
Y-coordinate: 8.00

Expected to display 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 getCounter(), expected to display 4
The number of points currently in existence are: 4

Displaying most recent point's id, expecting 1004
Unique id of Point p is: 1004
-------------------------------------------------------
Testing Functions in class Square:

Square Name: SQUARE - S
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 12
Area: 144
Perimeter: 48

Testing the move function, moving square a dx = 5, dy = -4
Expecting square to now be at coordinates x = 10.00, y = 3.00:

Square Name: SQUARE - S
X-coordinate: 10.00
Y-coordinate: 3.00
```

```
Side a: 12
Area: 144
Perimeter: 48


Testing the copy constructor

Expecting the following result for displaying sq1:
Square Name: SQUARE - S
X-coordinate: 10.00
Y-coordinate: 3.00
Side a: 12
Area: 144
Perimeter: 48

Square Name: SQUARE - S
X-coordinate: 10.00
Y-coordinate: 3.00
Side a: 12
Area: 144
Perimeter: 48

Testing the assignment operator:

Expecting the following result for displaying sq2:
Square Name: SQUARE - S
X-coordinate: 10.00
Y-coordinate: 3.00
Side a: 20
Area: 400
Perimeter: 80

Square Name: SQUARE - S
X-coordinate: 10.00
Y-coordinate: 3.00
Side a: 20
Area: 400
Perimeter: 80

------------------------------------------------------------

Testing Functions in class Rectangle:
```

```
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 12
Side b: 15
Area: 180
Perimiter: 54

Rectangle Name: RECTANGLE B
X-coordinate: 16.00
Y-coordinate: 7.00
Side a: 8
Side b: 9
Area: 72
Perimiter: 34

Distance between square a, and b is: 11

Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 12
Side b: 15
Area: 180
Perimiter: 54

Testing assignment operator in class Rectangle:

Rectangle Name: RECTANGLE rec2
X-coordinate: 3.00
Y-coordinate: 4.00
Side a: 11
Side b: 7
Area: 77
Perimiter: 36

Expected to display the following values for objec rec2:
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
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
Side b: 15
Area: 180
Perimiter: 54

Testing copy constructor in class Rectangle:
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 100
Side b: 200
Area: 20000
Perimiter: 600

Expected to display the following values for objec rec3:
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 100
Side b: 200
Area: 20000
Perimeter: 600
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 100
Side b: 200
Area: 20000
Perimiter: 600

--------------------------------------------------------
Testing array of pointers and polymorphism:
Square Name: SQUARE - S
X-coordinate: 10.00
Y-coordinate: 3.00
Side a: 10
```

```
Area: 100
Perimeter: 40
Rectangle Name: RECTANGLE B
X-coordinate: 16.00
Y-coordinate: 7.00
Side a: 8
Side b: 9
Area: 72
Perimiter: 34
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 12
Side b: 15
Area: 180
Perimiter: 54
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 100
Side b: 200
Area: 20000
Perimiter: 600

------------------------------------------------------------
```

Exercise B

Source Codes:

Class Circle

```
 /*
* File Name: circle.h
* Lab # and Assignment #: Lab 5 Exercise B
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/

#include "shape.h"

#ifndef CIRCLE_H
#define CIRCLE_H

class Circle: virtual public Shape {
    public:
        Circle(double x, double y, double radius, const char *name);
        //  REQUIRES: name is pointing to the first char of a built-in
string
        //  PROMISES: constructs a Circle object with its data member
initialized
        //  from the provided arguments

        const double getRadius() const {return radius;};
        //  PROMISES: returns the radius of the Circle

        void setRadius(double newRad) {radius = newRad;};
        //  PROMISES: sets the radius to the provided argument

        double area() const;
        //  PROMISES: returns the area of the Circle

        double perimeter() const;
        //  PROMISES: returns the perimeter of the Rectangle

        void display() const;
        //  PROMISES: display all data members to the console
    protected:
```

```
        double radius;
};

#endif
```

```
 /*
* File Name: circle.cpp
* Lab # and Assignment #: Lab 5 Exercise B
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/

#include "circle.h"
#include <iostream>
#include <math.h>

using namespace std;

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

double Circle::area() const {
    return 3.141592 * pow(getRadius(), 2);
}

double Circle::perimeter() const {
    return 2 * 3.141592 * getRadius();
}

void Circle::display() const {
    cout << "Circle Name: " << getName() << endl;
    getOrigin().display();
    cout << "Radius: " << getRadius() << endl;
    cout << "Area: " << area() << endl;
    cout << "Perimiter: " << perimeter() << endl;
}
```

Class CurveCut

```
/*
* File Name: curveCut.h
* Lab # and Assignment #: Lab 5 Exercise B
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/

#include "circle.h"
#include "rectangle.h"

#ifndef CURVECUT_H
#define CURVECUT_H

class CurveCut: public Rectangle, public Circle {
    public:
        CurveCut(double x, double y, double sideA, double sideB, double
radius, const char *name);
        //  REQUIRES:
        //      radius <= min(width, length)
        //      name is pointing to the first char of a built-in string
        //  PROMISES: constructs a CurveCut object with its data member
initialized
        //  from the provided arguments. Exits program if radius
conditions not met.

        double area() const;
        //  PROMISES: returns the area of the CurveCut

        double perimeter() const;
        //  PROMISES: returns the perimeter of the CurveCut

        void display() const;
        //  PROMISES: display all data members to the console
};

#endif
```

```
/*
* File Name: curveCut.cpp
* Lab # and Assignment #: Lab 5 Exercise B
```

```
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 23 2020
*/

#include "square.h"
#include "circle.h"
#include "curveCut.h"
#include <math.h>
#include <iostream>
#include <algorithm>

using namespace std;

CurveCut::CurveCut(double x, double y, double sideA, double sideB, double
radius, const char *name)
        : Shape(x, y, name), Rectangle(x, y, sideA, sideB, name),
Circle(x, y, radius, name) {
    if (radius > min(sideA, sideB)){
        cerr << "Error, radius is larger than min(width, length),
exiting..." << endl;
        exit(1);
    }
}

double CurveCut::area() const{
    return Rectangle::area() - (Circle::area() / 4);
}

double CurveCut::perimeter() const {
    return (Circle::perimeter() / 4) + (Rectangle::perimeter() - 2 *
Circle::getRadius());
}

void CurveCut::display() const{
    cout << "CurveCut name: " << getName() << endl;
    getOrigin().display();
    cout << "Width: " << getSideA() << endl;
    cout << "Length: " << getSideB() << endl;
    cout << "Radius of the cut: " << getRadius() << endl;
}
```

Program Output

Please refer to ExA graphicsWorld to see code for testing ExB

Only showing output for exercise B portion since Ex A output was already shown earlier in report

```
-------------------------------------------------------
Testing Functions in class Circle:
Circle Name: CIRCLE C
X-coordinate: 3.00
Y-coordinate: 5.00
Radius: 9
Area: 254.469
Perimiter: 56.5487

the area of CIRCLE C  is: 254.469
the perimeter of CIRCLE C is: 56.5487

The distance between rectangle a and circle c is: 2.82843


-------------------------------------------------------

Testing Functions in class CurveCut:

CurveCut name: CurveCut rc
X-coordinate: 6.00
Y-coordinate: 5.00
Width: 10
Length: 12
Radius of the cut: 9
the area of CurveCut rc is: 56.3828
the perimeter of CurveCut rc is: 40.1372

The distance between rc and c is: 3

Testing copy constructor in class CurveCut:

Expected to display the following values for objec cc:
Rectangle Name: CurveCurt rc
X-coordinate: 6.00
Y-coordinate: 5.00
Width: 12
Length: 10
```

```
Radius of the cut: 9

CurveCut name: CurveCut rc
X-coordinate: 6.00
Y-coordinate: 5.00
Width: 10
Length: 12
Radius of the cut: 9

Testing assignment operator in class CurveCut:
CurveCut name: CurveCut cc2
X-coordinate: 2.00
Y-coordinate: 5.00
Width: 100
Length: 12
Radius of the cut: 9

CurveCut name: CurveCut rc
X-coordinate: 6.00
Y-coordinate: 5.00
Width: 10
Length: 12
Radius of the cut: 9

------------------------------------------------------------
Testing array of pointers and polymorphism:
Square Name: SQUARE - S
X-coordinate: 10.00
Y-coordinate: 3.00
Side a: 10
Area: 100
Perimeter: 40

the area of SQUARE - S is: 100
the perimeter of SQUARE - S is: 40

Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 400
Side b: 300
Area: 120000
Perimiter: 1400
```

```
the area of RECTANGLE A is: 120000
the perimeter of RECTANGLE A is: 1400

Circle Name: CIRCLE C
X-coordinate: 3.00
Y-coordinate: 5.00
Radius: 9
Area: 254.469
Perimiter: 56.5487

the area of CIRCLE C is: 254.469
the circumference of CIRCLE C is: 56.5487

CurveCut name: CurveCut rc
X-coordinate: 6.00
Y-coordinate: 5.00
Width: 10
Length: 40
Radius of the cut: 9

the area of CurveCut rc is: 336.383
the perimeter of CurveCut rc is: 96.1372

------------------------------------------------------------
```