Course: ENSF 614 - Fall 2023

Lab #: Lab 5

Instructor: Prof. M. Moussavi

Student Name: Jeremy Sugimoto

Submission Date: Oct 23, 2023

Exercsie A:

point.h

```
// point.h
// ENSF 614 Fall 2023 Lab 5 - Exercise A

#ifndef POINT_H
#define POINT_H

class Point{
private:
    double x;
    double y;
    int id; // ID number

public:
    Point(double x, double y); // Constructor
    void display() const; // Display func
    double getX() const; // Getter for X
    double getY() const; // Getter for y
    int getId() const; // Getter for id
    void setX(double xSet); // Setter for X
    void setY(double ySet); // Setter for Y
    static int counter(); // Function to return the number of objects
    double distance(const Point& other) const; // Non-static distance function
    static double distance(const Point& p1, const Point& p2); // Static distance function


};
#endif
```

point.cpp

```
#include "point.h"
#include "point.h"  // Include the user-defined header file for the Point class
#include <iostream>  // Include the standard input/output stream library
#include <cmath>     // Include the math library for mathematical functions
using namespace std;  // Using the standard namespace

int count = 1000;  // Initialize a global count variable with the value 1000

// Constructor for the Point class, initializes x, y, and assigns a unique id
Point::Point(double x, double y): x(x), y(y){
    count++;  // Increment the global count variable
    id = count;  // Assign the unique id based on the global count
}

// Member function to display the coordinates of the point
void Point::display() const {
    cout << "X-coordinate: " << x << endl;
    cout << "Y-coordinate: " << y << endl;
}

// Getter method for retrieving the X-coordinate
double Point::getX() const {
    return x;
}

// Getter method for retrieving the Y-coordinate
double Point::getY() const {
    return y;
}

// Setter method to set the X-coordinate
void Point::setX(double xSet) {
    x = xSet;
}

// Setter method to set the Y-coordinate
void Point::setY(double ySet) {
    y = ySet;
}

// Getter method to retrieve the unique id of the point
int Point::getId() const {
    return id;
}

// Static member function to calculate the number of Point objects created
int Point::counter() {
    return count - 1000;
}

// Member function to calculate the Euclidean distance between two points
double Point::distance(const Point& other) const {
    double dx = x - other.x;
    double dy = y - other.y;
    return sqrt(dx * dx + dy * dy);
}

// Static member function to calculate the Euclidean distance between two points
double Point::distance(const Point& p1, const Point& p2) {
    double dx = p1.x - p2.x;
    double dy = p1.y - p2.y;
    return sqrt(dx * dx + dy * dy);
}
```

square.h

```
#ifndef SQUARE_H
#define SQUARE_H

```

```
#include "shape.h"  // Include the definition of the base class "Shape"

class Square : virtual public Shape {  // Define the Square class that inherits from Shape
private:
    double side_a;  // Private member variable to store the side length of the square

public:
    // Constructor for the Square class
    Square(double x, double y, double side_a, const char* name);

    // Override the base class's virtual function to calculate the area
    double area() const override;

    // Override the base class's virtual function to calculate the perimeter
    double perimeter() const override;

    // Getter method to retrieve the value of side_a
    double getSideA() const;

    // Setter method to set the value of side_a
    void setSideA(double side_a);

    // Override the base class's virtual function to display square information
    void display() const override;
};

#endif
```

square.cpp

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

// Constructor for the Square class, initializing its members
Square::Square(double x, double y, double side_a, const char* name) : Shape(x, y, name),
side_a(side_a) {
    // The constructor initializes the base class's members using the base class constructor
}

// Calculate and return the area of the square
double Square::area() const {
    return side_a * side_a;
}

// Calculate and return the perimeter of the square
double Square::perimeter() const {
    return 4 * side_a;
}
```

```cpp
// Getter method to retrieve the value of the side length (side_a)
double Square::getSideA() const {
    return side_a;
}

// Setter method to set the value of the side length (side_a)
void Square::setSideA(double side_a) {
    this->side_a = side_a;
}

// Display information about the square, including its base class information
void Square::display() const {
    Shape::display();  // Call the display method of the base class (Shape)
    cout << "Side a: " << side_a << endl;
    cout << "Area: " << area() << endl;
    cout << "Perimeter: " << perimeter() << endl;
}
```

rectangle.h

```cpp
#ifndef RECTANGLE_H
#define RECTANGLE_H

#include "shape.h"

class Rectangle : virtual public Shape{
private:
    double side_a;
    double side_b;

public:
    Rectangle(double x, double y, double side_a, double side_b, const char* name); //
Constructor
    double area() const override; // Calculate and return the area of a rectangle
    double perimeter() const override; // Calculate and return the perimeter of a rectangle
    double getSideA() const; // Getter for side_a
    void setSideA(double side_a); // Setter for side_a
    double getSideB() const; // Getter for side_b
    void setSideB(double side_b); // Setter for side_b
    void display() const override; // Display the properties of the rectangle
};

#endif
```

rectangle.cpp

```cpp
#include "rectangle.h"
#include <iostream>
using namespace std;

// Constructor for the Rectangle class, initializing its members
```

```cpp
Rectangle::Rectangle(double x, double y, double side_a, double side_b, const char* name)
    : Shape(x, y, name), side_a(side_a), side_b(side_b) {
    // The constructor initializes the base class's members using the base class constructor
}

// Calculate and return the area of the rectangle
double Rectangle::area() const {
    return side_a * side_b;
}

// Calculate and return the perimeter of the rectangle
double Rectangle::perimeter() const {
    return 2 * (side_a + side_b);
}

// Getter method to retrieve the value of side_a (width)
double Rectangle::getSideA() const {
    return side_a;
}

// Setter method to set the value of side_a (width)
void Rectangle::setSideA(double side_a) {
    this->side_a = side_a;
}

// Getter method to retrieve the value of side_b (height)
double Rectangle::getSideB() const {
    return side_b;
}

// Setter method to set the value of side_b (height)
void Rectangle::setSideB(double side_b) {
    this->side_b = side_b;
}

// Display information about the rectangle, including its base class information
void Rectangle::display() const {
    Shape::display();  // Call the display method of the base class (Shape)
    cout << "Side a (width): " << side_a << endl;
    cout << "Side b (height): " << side_b << endl;
    cout << "Area: " << area() << endl;
    cout << "Perimeter: " << perimeter() << endl;
    cout << endl;  // Add an extra line for spacing
}
```

graphicsWorld.h

```cpp
#ifndef GRAPHICSWORLD_H
#define GRAPHICSWORLD_H

#include "point.h"      // Include the header file for the "Point" class
#include "shape.h"      // Include the header file for the "Shape" class
```

```cpp
#include "square.h"     // Include the header file for the "Square" class
#include "rectangle.h"  // Include the header file for the "Rectangle" class
#include "circle.h"     // Include the header file for the "Circle" class
#include "curveCut.h"   // Include the header file for the "CurveCut" class

class GraphicsWorld {
public:
    // Declaration of the run function, which likely contains the main logic of the graphics
application
    void run();
};

#endif
```

graphicsWorld.cpp

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

void GraphicsWorld::run() {
#if 1
 // Change 0 to 1 to test Point
    cout << "Testing Functions in class Point:" << endl;
    Point m(6, 8);
    Point n(6, 8);
    n.setX(9);
    cout << "Expected to display the distance between m and n is: 3" << endl;
    cout << "The distance between m and n is: " << m.distance(n) << endl;
    cout << "Expected second version of the distance function also print: 3" << endl;
    cout << "The distance between m and n is again: " << Point::distance(m, n) << endl;
#endif // end of block to test Point

#if 1 // Change 0 to 1 to test Square
    cout << "\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:" << endl;
    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 << "Distance between rectangle a and b is: " << d << endl << endl;

    Rectangle rec1 = a;
    rec1.display();

    cout << "Testing assignment operator in class Rectangle:" << endl;
    Rectangle rec2(3, 4, 11, 7, "RECTANGLE rec2");
    rec2.display();
    rec2 = a;
    a.setSideB(200);
    a.setSideA(100);
    cout << "Expected to display the following values for object 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 << "Testing copy constructor in class Rectangle:" << endl;
    Rectangle rec3(a);
    rec3.display();
    a.setSideB(300);
    a.setSideA(400);
    cout << "Expected to display the following values for object rec3: " << 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 copy constructor." << endl;
    rec3.display();

#endif // end of block to test Rectangle

#if 1 // Change 0 to 1 to test using an array of pointers and polymorphism
    cout << "\nTesting array of pointers and polymorphism:" << endl;
    Shape* sh[4];
    sh[0] = &s;
    sh[1] = &b;
    sh[2] = &a;
    sh[3] = &rec3;
    for (int i = 0; i < 4; i++) {
        sh[i]->display();
    }
    cout << endl << endl;
```

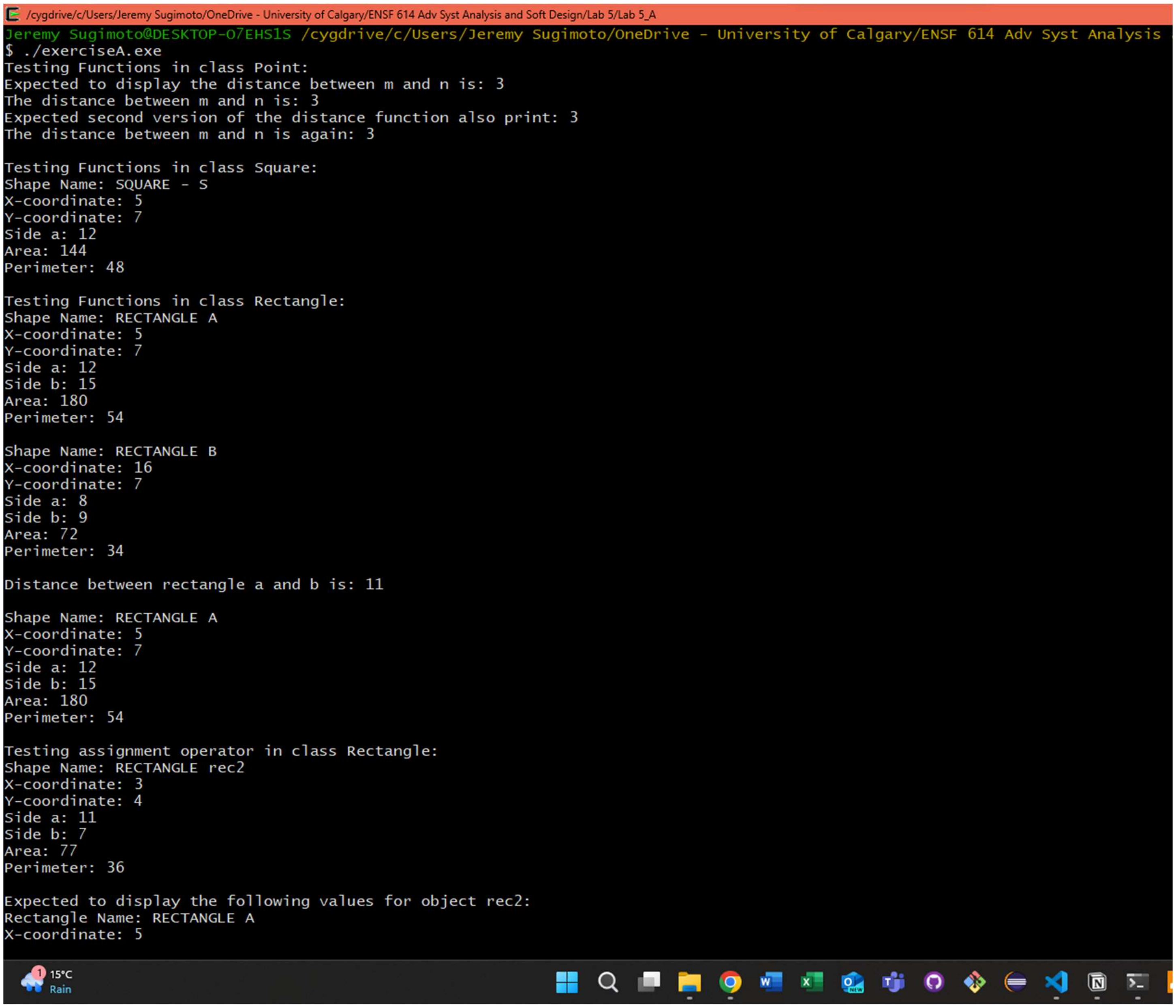
```
#endif // end of block to test array of pointers and polymorphism

#if 0
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;
CurveCut rc (6, 5, 10, 12, 4, "CurveCut rc");
rc.display();
cout << "the area of " << rc.getName() <<" is: "<< rc.area();
cout << "the perimeter of " << rc.getName() << " is: "<< rc.perimeter();
d = rc.distance(c);
cout << "\nThe distance between rc and c is: " <<d;

// 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
} // END OF FUNCTION run
```

lab5ExA.cpp

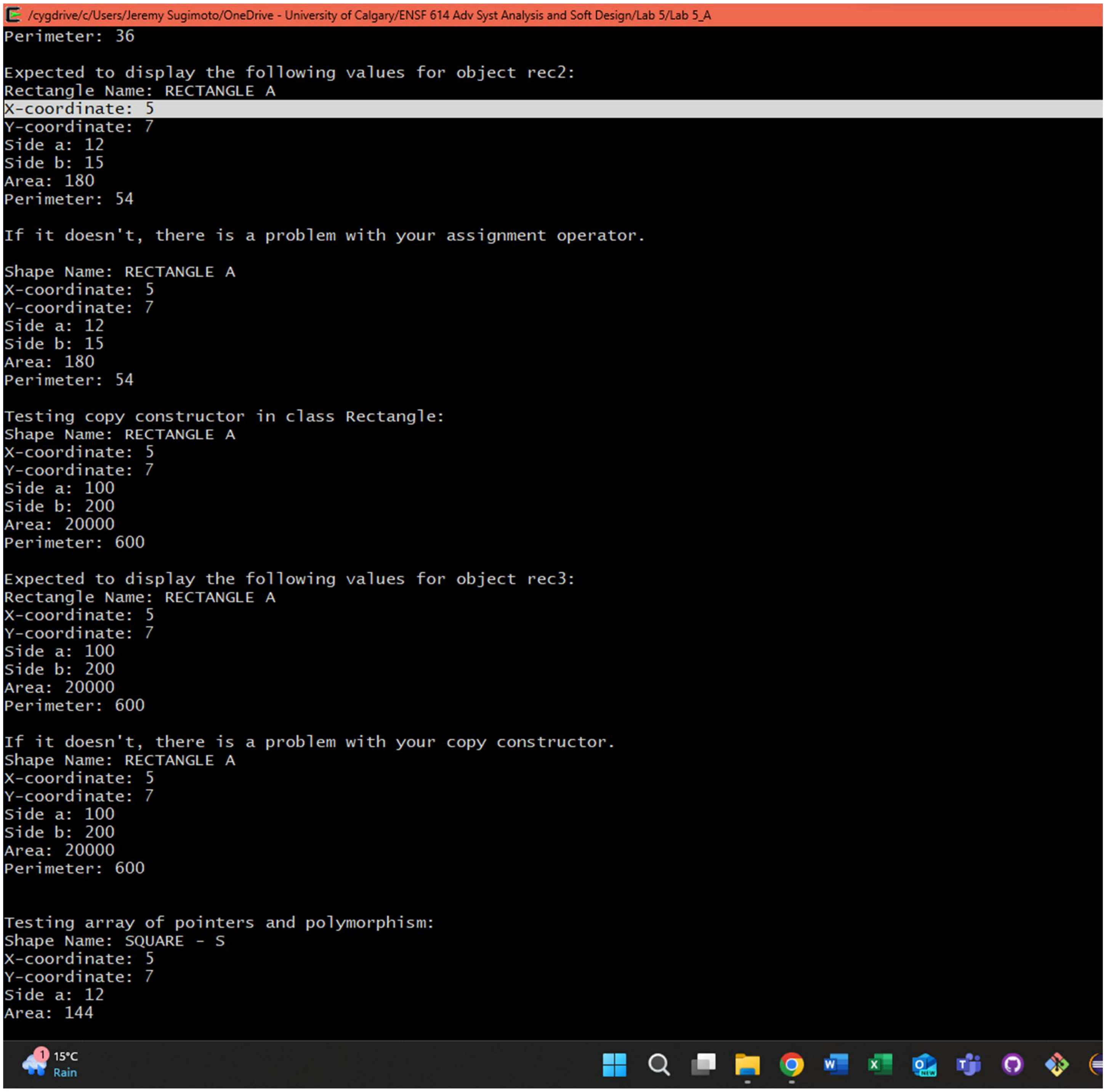
```
// lab5_exeA
// ENSF 614 Fall 2022 LAB 5 - EXERCISE A
```

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

#include "graphicsWorld.h"
#include "point.h"
#include "shape.h"
#include "square.h"
#include "rectangle.h"
#include "circle.h"
#include "curveCut.h"



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

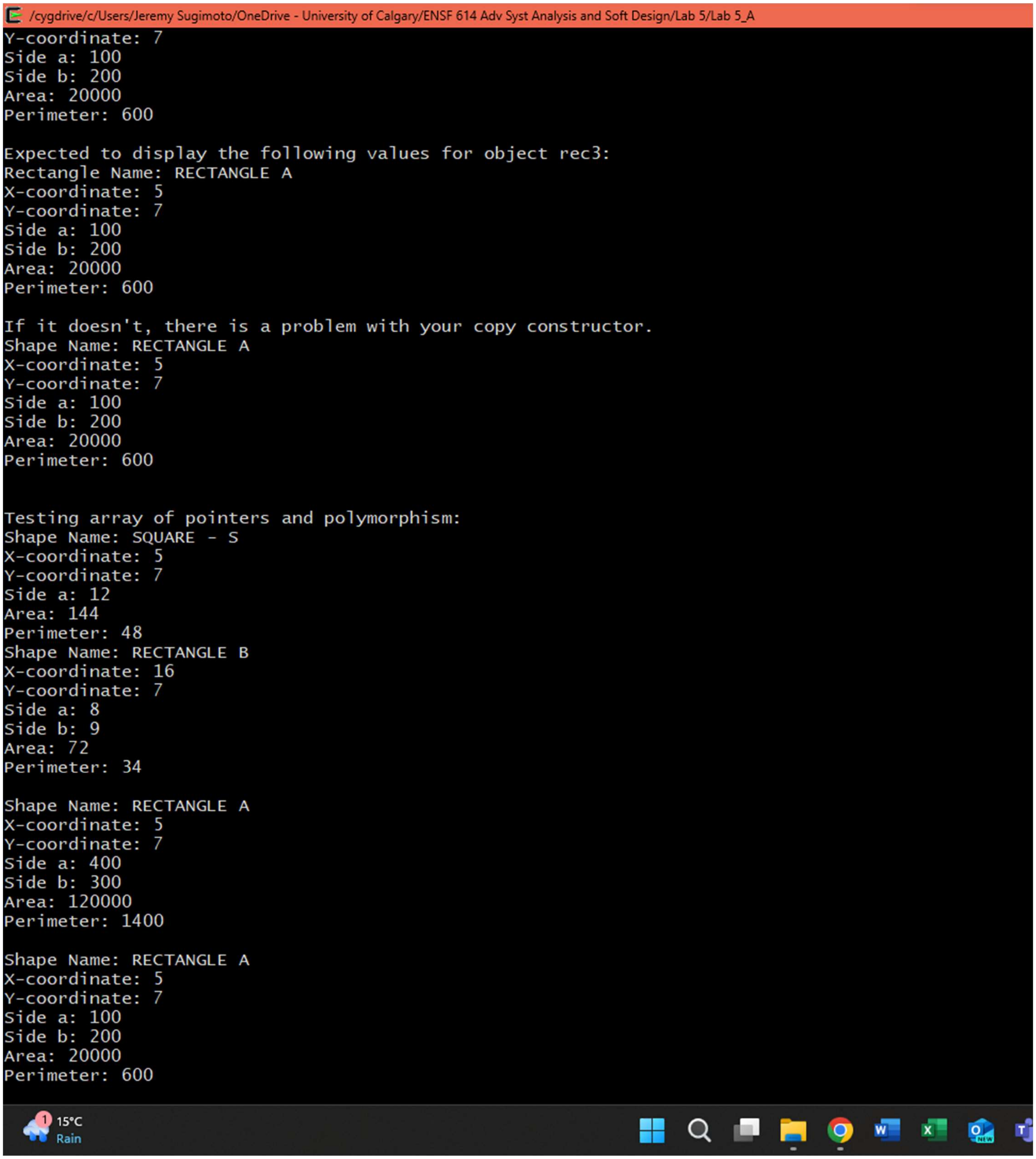
Program Output Part A



```
/cygdrive/c/Users/Jeremy Sugimoto/OneDrive - University of Calgary/ENSF 614 Adv Syst Analysis and Soft Design/Lab 5/Lab 5_A
Jeremy Sugimoto@DESKTOP-O7EHS1S /cygdrive/c/Users/Jeremy Sugimoto/OneDrive - University of Calgary/ENSF 614 Adv Syst Analysis
$ ./exerciseA.exe
Testing Functions in class Point:
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 Functions in class Square:
Shape Name: SQUARE - S
X-coordinate: 5
Y-coordinate: 7
Side a: 12
Area: 144
Perimeter: 48

Testing Functions in class Rectangle:
Shape Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Side a: 12
Side b: 15
Area: 180
Perimeter: 54

Shape Name: RECTANGLE B
X-coordinate: 16
Y-coordinate: 7
Side a: 8
Side b: 9
Area: 72
Perimeter: 34

Distance between rectangle a and b is: 11

Shape Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Side a: 12
Side b: 15
Area: 180
Perimeter: 54

Testing assignment operator in class Rectangle:
Shape Name: RECTANGLE rec2
X-coordinate: 3
Y-coordinate: 4
Side a: 11
Side b: 7
Area: 77
Perimeter: 36

Expected to display the following values for object rec2:
Rectangle Name: RECTANGLE A
X-coordinate: 5
```


/cygdrive/c/Users/Jeremy Sugimoto/OneDrive - University of Calgary/ENSF 614 Adv Syst Analysis and Soft Design/Lab 5/Lab 5_A

```
Perimeter: 36

Expected to display the following values for object 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.

Shape Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Side a: 12
Side b: 15
Area: 180
Perimeter: 54

Testing copy constructor in class Rectangle:
Shape Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Side a: 100
Side b: 200
Area: 20000
Perimeter: 600

Expected to display the following values for object rec3:
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 copy constructor.
Shape Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Side a: 100
Side b: 200
Area: 20000
Perimeter: 600


Testing array of pointers and polymorphism:
Shape Name: SQUARE - S
X-coordinate: 5
Y-coordinate: 7
Side a: 12
Area: 144
```


/cygdrive/c/Users/Jeremy Sugimoto/OneDrive - University of Calgary/ENSF 614 Adv Syst Analysis and Soft Design/Lab 5/Lab 5_A
15°C
Rain

```
Y-coordinate: 7
Side a: 100
Side b: 200
Area: 20000
Perimeter: 600

Expected to display the following values for object rec3:
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 copy constructor.
Shape Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Side a: 100
Side b: 200
Area: 20000
Perimeter: 600


Testing array of pointers and polymorphism:
Shape Name: SQUARE - S
X-coordinate: 5
Y-coordinate: 7
Side a: 12
Area: 144
Perimeter: 48
Shape Name: RECTANGLE B
X-coordinate: 16
Y-coordinate: 7
Side a: 8
Side b: 9
Area: 72
Perimeter: 34

Shape Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Side a: 400
Side b: 300
Area: 120000
Perimeter: 1400

Shape Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Side a: 100
Side b: 200
Area: 20000
Perimeter: 600
```

Exercise B:

circle.h

```
#ifndef CIRCLE_H
#define CIRCLE_H

#include "shape.h"
```

```cpp
class Circle : virtual public Shape {
private:
    double radius;

public:
    // Constructor
    Circle(double x, double y, double radius, const char* name);

    // Override Area calculation function
    double area() const override;

    // Overrdie Perimeter calculation function (Circumference)
    double perimeter() const override;

    // Getter and Setter for radius
    double getRadius() const;
    void setRadius(double newRadius);

    // Override Display function
    void display() const override;
};

#endif
```

circle.cpp

```cpp
#include "circle.h"
#include <iostream>
#include <cmath>
using namespace std;

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

// Area calculation function
double Circle::area() const {
    return M_PI * radius * radius;
}

// Perimeter calculation function (Circumference)
double Circle::perimeter() const {
    return 2 * M_PI * radius;
}

// Getter for radius
double Circle::getRadius() const {
    return radius;
}
```

```
// Setter for radius
void Circle::setRadius(double newRadius) {
    radius = newRadius;
}

// Display function
void Circle::display() const {
    Shape::display();
    cout << "Radius: " << radius << endl;
    cout << "Area: " << area() << endl;
    cout << "Perimeter: " << perimeter() << endl;
}
```

curveCut.h

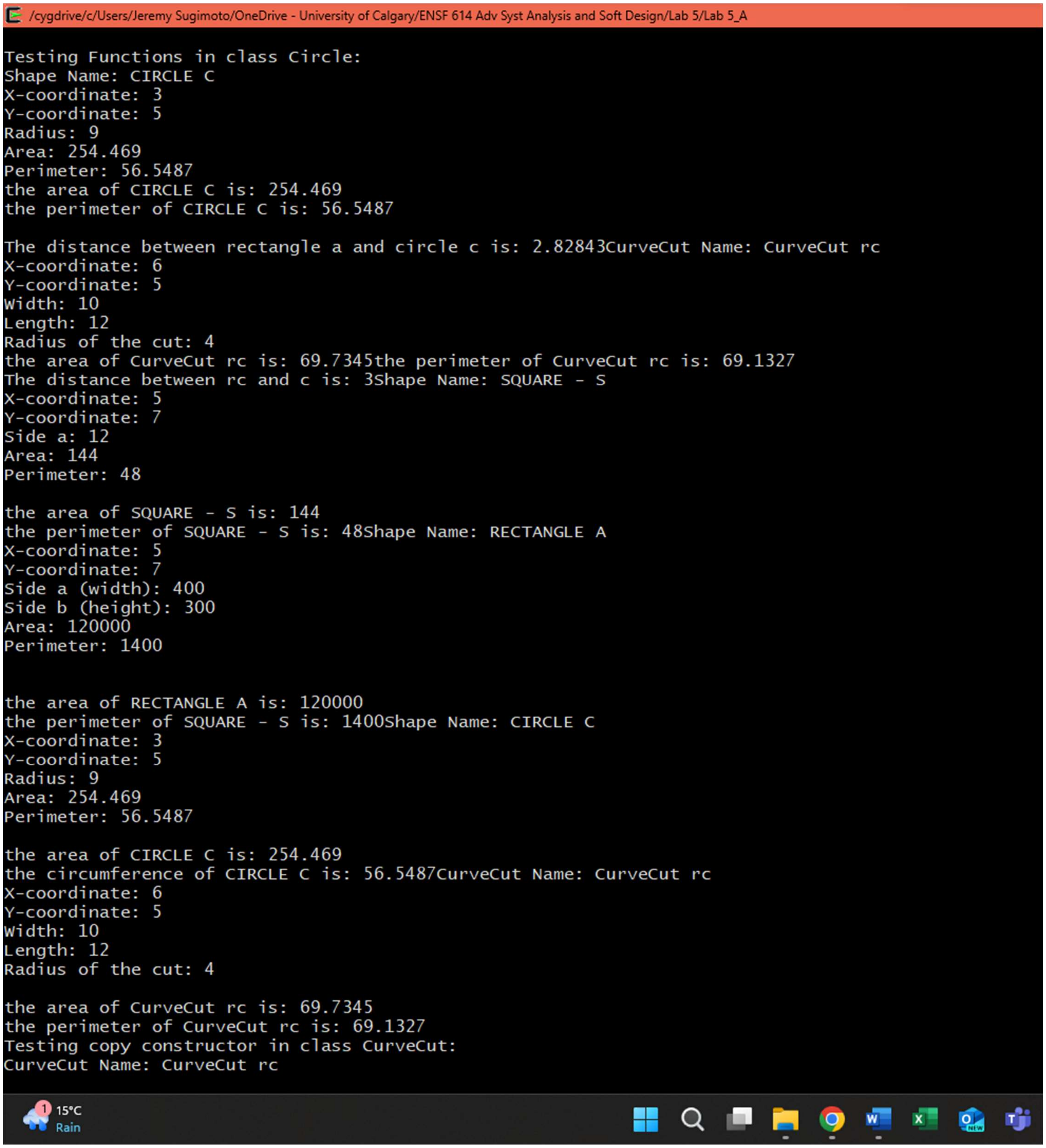
```
#ifndef CURVECUT_H
#define CURVECUT_H

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

class CurveCut : public Rectangle, public Circle{
public:
    // Constructor
    CurveCut(double x, double y, double width, double length, double radius, const char*
name);

    // Override area function
    double area() const override;

    // Override perimeter function
    double perimeter() const override;

    // Override display function
    void display() const override;
};

#endif
```

curveCut.cpp

```
#include "curveCut.h"
#include <iostream>
#include <cmath>
using namespace std;

// Constructor
CurveCut::CurveCut(double x, double y, double width, double length, double radius, const
char* name)
    : Shape(x, y, name), Rectangle(x, y, width, length, name), Circle(x, y, radius, name) {
    if (radius > width / 2.0 || radius > length / 2.0) {
```

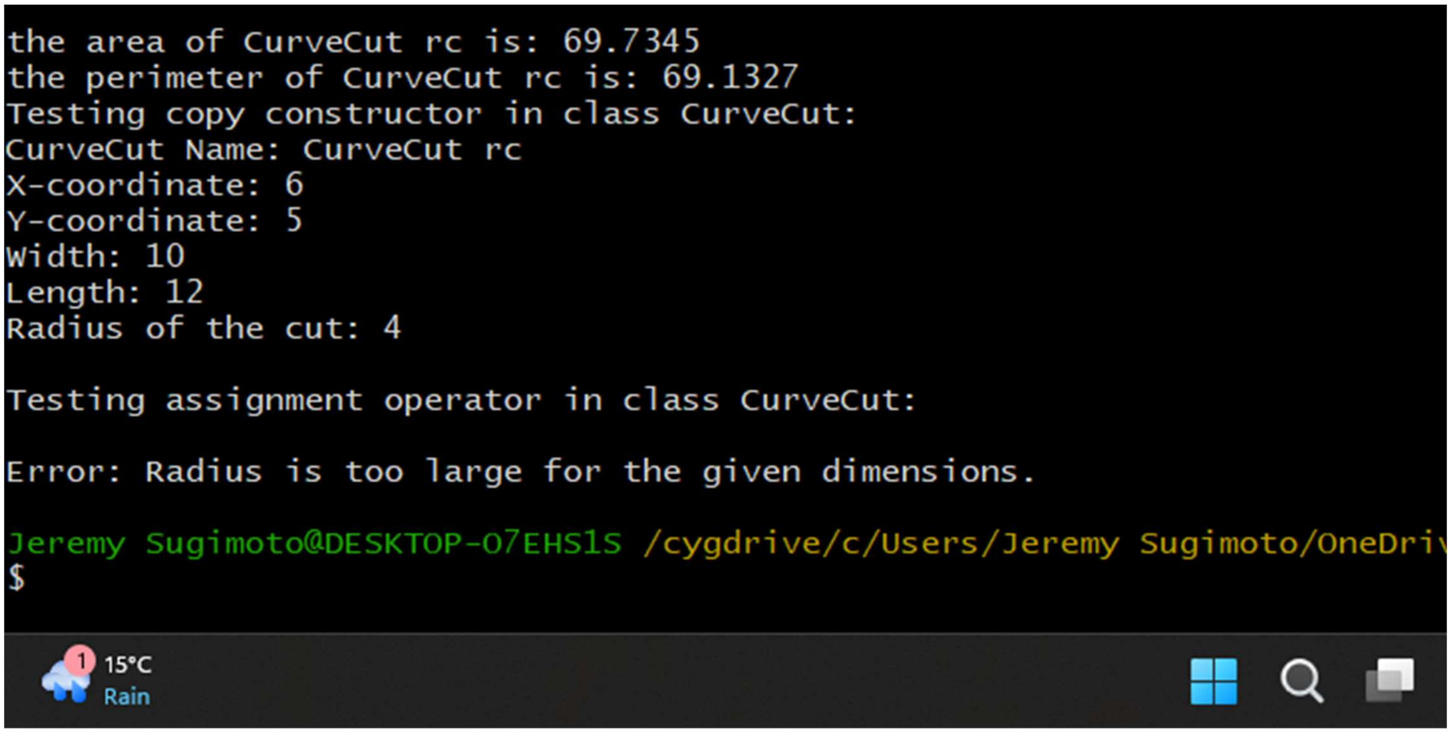
```cpp
        cerr << "\nError: Radius is too large for the given dimensions." << endl;
        exit(1);
    }
}

// Override area function
double CurveCut::area() const {
    double rectArea = Rectangle::area();
    double circleArea = Circle::area();
    return rectArea - circleArea;
}

// Override perimeter function
double CurveCut::perimeter() const {
    double rectPerimeter = Rectangle::perimeter();
    double circlePerimeter = Circle::perimeter();
    return rectPerimeter + circlePerimeter;
}

// Override display function
void CurveCut::display() const {
    cout << "CurveCut Name: " << Rectangle::getName() << endl;
    cout << "X-coordinate: " << Rectangle::getOrigin().getX() << endl;
    cout << "Y-coordinate: " << Rectangle::getOrigin().getY() << endl;
    cout << "Width: " << Rectangle::getSideA() << endl;
    cout << "Length: " << Rectangle::getSideB() << endl;
    cout << "Radius of the cut: " << Circle::getRadius() << endl;
}
```

Program Output Exercise B


/cygdrive/c/Users/Jeremy Sugimoto/OneDrive - University of Calgary/ENSF 614 Adv Syst Analysis and Soft Design/Lab 5/Lab 5_A

```
Testing Functions in class Circle:
Shape Name: CIRCLE C
X-coordinate: 3
Y-coordinate: 5
Radius: 9
Area: 254.469
Perimeter: 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: 2.82843CurveCut Name: CurveCut rc
X-coordinate: 6
Y-coordinate: 5
Width: 10
Length: 12
Radius of the cut: 4
the area of CurveCut rc is: 69.7345the perimeter of CurveCut rc is: 69.1327
The distance between rc and c is: 3Shape Name: SQUARE - S
X-coordinate: 5
Y-coordinate: 7
Side a: 12
Area: 144
Perimeter: 48

the area of SQUARE - S is: 144
the perimeter of SQUARE - S is: 48Shape Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Side a (width): 400
Side b (height): 300
Area: 120000
Perimeter: 1400


the area of RECTANGLE A is: 120000
the perimeter of SQUARE - S is: 1400Shape Name: CIRCLE C
X-coordinate: 3
Y-coordinate: 5
Radius: 9
Area: 254.469
Perimeter: 56.5487

the area of CIRCLE C is: 254.469
the circumference of CIRCLE C is: 56.5487CurveCut Name: CurveCut rc
X-coordinate: 6
Y-coordinate: 5
Width: 10
Length: 12
Radius of the cut: 4

the area of CurveCut rc is: 69.7345
the perimeter of CurveCut rc is: 69.1327
Testing copy constructor in class CurveCut:
CurveCut Name: CurveCut rc
```

15°C
Rain



```
the area of CurveCut rc is: 69.7345
the perimeter of CurveCut rc is: 69.1327
Testing copy constructor in class CurveCut:
CurveCut Name: CurveCut rc
X-coordinate: 6
Y-coordinate: 5
Width: 10
Length: 12
Radius of the cut: 4

Testing assignment operator in class CurveCut:

Error: Radius is too large for the given dimensions.

Jeremy Sugimoto@DESKTOP-O7EHS1S /cygdrive/c/Users/Jeremy Sugimoto/OneDri
$
```