ENSF 619 – Fall 2020

Ziad Chemali

Lab # 5

October 23,2020

Exercise: A

1) Code:

i) Header files:

graphicWorld.h

```
/*
*File Name: Exercise_A, graphicsWorld.h
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/

#ifndef graphics_world
#define graphics_world
class GraphicsWorld {
public :
	//PROMISES: Test single inheritance
	void run();

};
#endif
```

point.h

```
/*
*File Name: Exercise_A, point.h
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/
#ifndef point_h
#define point_h

class Point {

public:
	/*
	*PROMISES:  static variable to keep track of how many Point objects are created
	*/
	static int counter;

	/*
	* PROMISES: This function returns the distance between two point object
	*/
	static double distance(const Point& a, const Point& b);


	/*
	* PROMISES: returns the static variable counter
	*/
	static int get_counter();
```

```
        /*
        * PROMISES: displays the x,y coordinates of this Object
        */
        void display() const;

        /*
        * PROMISES: constructor that sets the x,y private variables
        */
        Point(double x, double y);

        /*
        * PROMISES: returns x
        */
        double getx() const;

        /*
        * PROMISES: returns y
        */
        double gety() const;

        /*
        * PROMISES: sets x
        */
        void setx(double x) ;

        /*
        * PROMISES: sets y
        */
        void sety(double y);

        /*
        * PROMISES: returns id
        */
        int get_id() const;

        /*
        * PROMISES: overloads assignment operator of Point
        */
        Point& operator=(const Point& rhs);

        /*
        * PROMISES: copy constructor
        */
        Point(const Point& r);

        /*
        * PROMISES: destructor that decrements counter hen Point is deleted
        */
        ~Point();

        /*
        * PROMISES: returns the distance between this Point and other Point object
        */
        double distance(const Point& a);

private:
        double x;
```

```
	double y;
	int id;

};

#endif
```

rectangle.h

```
/*
*File Name: Exercise_A, Rectangle.h
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/


#include "square.h"
#ifndef rectangle_h
#define rectangle_h
class Rectangle : public Square {
public:
	/*
	* PROMISES: Constructs the Rectangle object and invokes the Square constructor
	*/
	Rectangle(double x, double y, double side_a, double side_b, const char* name);

	/*
	* PROMISES: Calculates the area of Rectangle object
	*/
	double area() const;
	double get_side_b() const;
	void set_side_b(double num);
	double perimeter() const;
	void display();
	Rectangle(const Rectangle& r);
	Rectangle& operator=( Rectangle& rhs);
private:
	double side_b;
};
#endif // !rectangle_h
```

Shape.h

```
/*
*File Name: Exercise_A, Shape.h
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/


#include "Point.h"
```

```
#include<iostream>
using namespace std;
#ifndef shape_h
#define shape_h
class Shape {
public:
	/*
	* PROMISES: Constructor that invokes the Point constructor and sets shapeName
dynamically
	*/
	Shape(double x, double y,const char* name);

	/*
	* PROMISES: returns the distance between two shapes
	*/
	static double distance(Shape& the_shape, Shape& other);


	/*
	* PROMISES: returns the distance between this and another shape object
	*/
	double distance(Shape& other);

	/*
	* PROMISES: destructor that deletes the shapeName
	*/
	virtual~Shape();

	/*
	* PROMISES: copy constructor
	*/
	Shape(const Shape& r);

	/*
	* PROMISES: overloading  assignmnet operator
	*/
	Shape& operator=(const Shape& rhs);

	/*
	* PROMISES: returns counter of Point object
	*/
	int get_counter() const;

	/*
	* PROMISES: returns id of Point object
	*/
	int get_id() const;

	/*
	* PROMISES: displays the name and coordinates od Shape
	*/
	void display() const;

protected:
	/*
	* PROMISES: returns Point object
	*/
	const Point& getOrigin();
	/*
	* PROMISES: returns name
	*/
	const char* getName() const;

	/*
	* PROMISES: moves the x,y coordinates by dx and dy
	*/
	void move(double dx, double dy);
private:
	Point origin;
	char* shapeName;
};

#endif // !shape.h
```

Square.h

```
 /*
*File Name: Exercise_A, Square.h
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/

#include "shape.h"
#include<iostream>
using namespace std;
#ifndef square_h
#define square_h
class Square: public Shape{

public:
	/*
	* PROMISES: sets the side private variable and invokes the constructor of Shape
	*/
	Square(double x, double y, double side, const char* name);

	/*
	* PROMISES: returns the area of square object
	*/
	double area() const;

	/*
	* PROMISES: returns the perimeter of square object
	*/
	double perimeter() const;

	/*
	* PROMISES: displays the name,x,y coordiantes and squares side
	*/
	void display();

	/*
	* PROMISES: returns side
```

```
        */
        double get_side_a() const;

        /*
        * PROMISES: sets side
        */
        void set_side_a(double num);

        /*
        * PROMISES: copy constructor of Square
        */
        Square(const Square& r);

        /*
        * PROMISES: overloads assignmnet operator
        */
        Square& operator=(const Square& rhs);

private:
        double side;

};
#endif // !square_h
```

ii) Source files

App.cpp

```
/*
*File Name: Exercise_A, app.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/
#include "graphicsWorld.h"
int main() {
        GraphicsWorld test;
        test.run();
}
```

graphicsWorld.cpp

```
/*
*File Name: Exercise_A, graphicsWorld.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/
#include"graphicsWorld.h"
#include "Point.h"
#include "rectangle.h"
#include "square.h"
#include<iostream>
using namespace std;

void GraphicsWorld::run() {
        //Exercise A-- Test
        cout << "Testing Single Inheritance Exercise, completed by Ziad Chemali" << endl;
```

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

       Rectangle b(16, 7, 8, 9, "RECTANGLE B");

       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* 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();
#endif



}
```

Point.cpp

```
/*
*File Name: Exercise_A, point.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/


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

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

double Point::distance(const Point& a)
{
	return sqrt(pow((x - a.x), 2) + pow((y - a.y), 2));
}

double Point::distance(const Point& a, const Point& b)
{
	return sqrt(pow((b.x-a.x),2)+ pow((b.y - a.y), 2));
}

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

void Point::display() const
{
	cout << "X-Coordinate: " << setprecision(8) << this->x<<endl;
	cout << "Y-Coordinate: " << setprecision(8) << this->y << endl;

}
```

```
Point::Point(double x=0, double y=0)
{
	this->x = x;
	this->y = y;
	counter++;
	id = 1001 +counter;
}

double Point::getx() const
{
	return x;
}

double Point::gety() const
{
	return y;
}

void Point::setx(double x)
{
	this->x = x;
}

void Point::sety(double y)
{
	this->y = y;

}

int Point::get_id() const
{
	return id;
}

Point& Point::operator=(const Point& rhs)
{
	if (this != &rhs) {
		this->x = rhs.getx();
		this->y = rhs.gety();
		this->id = rhs.id;

	}
	return *this;
}

Point::Point(const Point& r)
{
	this->x = r.getx();
	this->y = r.gety();
	this->id = r.id;
}


int Point::counter = 0;
```

rectangle.cpp

```
/*
```

```
*File Name: Exercise_A, rectangle.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/




#include "rectangle.h"
#include "shape.h"
#include<iomanip>
double Rectangle::area() const
{
	return side_b*get_side_a();
}

double Rectangle::perimeter() const
{
	return side_b*2+get_side_a()*2;
}

void Rectangle::display()
{
	Shape::display();
	cout << "side a: " <<setprecision(9)<<get_side_a()<< endl;
	cout << "side b: "<<setprecision(9) << get_side_b() << endl;
	cout << "Area: " << setprecision(9)<<area()<< endl;
	cout << "Perimeter: " << setprecision(9) << perimeter() << endl;
}

Rectangle::Rectangle(const Rectangle& r):Square(r)
{
	side_b = r.side_b;

}

Rectangle& Rectangle::operator=( Rectangle& rhs)
{

	if (this != &rhs) {
		Square::operator=(rhs);
		side_b = rhs.side_b;

	}
	return *this;
}

double Rectangle::get_side_b() const
{
	return side_b;
}

void Rectangle::set_side_b(double num)
{
	this->side_b = num;
}
```

```
Rectangle::Rectangle(double x, double y, double side_a, double side_b, const char* name)
:Square(x, y, side_a, name) {
	this->side_b = side_b;
}
```

Shape.cpp

```
/*
*File Name: Exercise_A, shape.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/



#include "shape.h"
#include "Point.h"
#include <string>
#include <cassert>

Shape::Shape(double x, double y, const char* name) :origin(x, y)
{
	const char* temp = name;
	int n = 0;
	while (*temp) {
		n++;
		temp++;
	}
	if (n > 0) {

	this->shapeName = new char[n+1];//to include \0

	for (int i = 0;i < n;i++) {
		shapeName[i] = name[i];
	}
	shapeName[n] = '\0';
}
	else {
		cout << "Name parameter is empty" << endl;
	}
}

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

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

Shape::Shape(const Shape& r):origin(r.origin)
{

	delete[] shapeName;
	const char* temp = r.getName();
	int n = 0;
	while (*temp) {
		n++;
		temp++;
	}
	if (n > 0) {

		this->shapeName = new char[n + 1];//to include \0

		for (int i = 0;i < n;i++) {
			shapeName[i] = r.getName()[i];
		}
		shapeName[n] = '\0';

	}

}

Shape& Shape::operator=(const Shape& rhs)
{
	if (this!= &rhs) {
		origin = rhs.origin;
		delete[] shapeName;
		const char* temp = rhs.getName();
		int n = 0;
		while (*temp) {
			n++;
			temp++;
		}
		if (n > 0) {

			this->shapeName = new char[n + 1];//to include \0

			for (int i = 0;i < n;i++) {
				shapeName[i] = rhs.getName()[i];
			}
			shapeName[n] = '\0';

		}

	}
	return *this;
}

int Shape::get_counter() const
{
	return this->origin.get_counter();
}

int Shape::get_id() const
{
	return origin.get_id();
}

const Point& Shape::getOrigin()
{
	return  origin;
}
```

```
const char* Shape::getName() const
{
	return shapeName;
}



void Shape::display() const
{

	cout << "Shape name: "<< shapeName << endl;
	origin.display();

}

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

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

Square.cpp

```
/*
*File Name: Exercise_A, Square.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/


#include "square.h"
#include <iomanip>
Square::Square(double x, double y, double side, const char* name):Shape(x, y, name)
{
	this->side = side;

}

double Square::get_side_a() const
{
	return side;
}

void Square::set_side_a(double num)
{
	side = num;
}

Square::Square(const Square& r):Shape(r)
{

	this->set_side_a(r.get_side_a());
}

Square& Square::operator=(const Square& rhs)
{
	if (this != &rhs) {
		Shape::operator=(rhs);
		this->side=rhs.get_side_a();

	}
	return *this;
}

double Square::area() const
{
	return pow(side,2);
}

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

void Square::display()
{
	Shape::display();
	cout << "side a: " << setprecision(8) << side << endl;
	cout << "Area: " <<setprecision(8) <<area()<< endl;
	cout << "Perimeter: " << setprecision(8) << perimeter() << endl;
}
```

2) Code Output:

```
Testing Single Inheritance Exercise, completed by Ziad Chemali


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:

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

Distance between square 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 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.

```
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 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.

```
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
Shape name: RECTANGLE B
X-Coordinate: 16
Y-Coordinate: 7
Shape name: RECTANGLE A
X-Coordinate: 5
Y-Coordinate: 7
Shape name: RECTANGLE A
X-Coordinate: 5
Y-Coordinate: 7
```

Exercise: B

1) Code:

i) Header files:

Circle.h

```
/*
*File Name: Exercise_B,circle.h
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/


#include"shape.h"
# ifndef PI
#define PI 3.14159265358979323846
#endif
#ifndef circle_h
#define circle_h

class Circle :public virtual Shape {
public:
       /*
       * PROMISES: constructor for Circle that invokes Shape constructor
       */
       Circle(double x, double y, double r,  const char* name);

       /*
       * Overriding pure virtual area function in Shape class,
       *  PROMISES: returns the area of circle
       */
       double area() const override;

       /*
       * PROMISES: return radius
       */
       double get_radius() const;

       /*
       * PROMISES: sets radius
       */
       void set_radius(double num);

       /*
       * Overriding pure virtual area function in Shape class,
       *  PROMISES: returns the perimeter of circle
       */
       double perimeter() const override;
       /*
       * Overriding pure virtual area function in Shape class,
       *  PROMISES: displays name, coordinates,radius, area, and perimeter of Circle
       */
       void display() override;
```

```
        /*
        * PROMISES: copy constructor of Circle
        */
        Circle (const Circle& r);

        /*
        * PROMISES: Overloads assignment operator of Circle
        */
        Circle& operator=(Circle& rhs);
private:
        double radius;
};
#endif // !circle_h
```

curveCut.h

```
/*
*File Name: Exercise_B,curveCut.h
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/



#include"rectangle.h"
#include"circle.h"
#ifndef CurveCut_h
#define CurveCut_h

class CurveCut :public Rectangle, public Circle {
public:
        /*
        * REQUIRES: radius <= min of(length,width)
        * PROMISES: constructs CurveCut and invokes Shape, Circle, Rectangle constructors
        */
        CurveCut(double x, double y, double side_a, double side_b, double radius, const
char* name);

        /*
        * Overrides pure virtual function in Shape class
        * PROMISES: returns area of CurveCut
        */
        double area() const override;

        /*
        * Overrides pure virtual function in Shape class
        * PROMISES: returns perimeter of CurveCut
        */
        double perimeter() const override;

        /*
        * Overrides pure virtual function in Shape class
        * PROMISES: displays name, coordinates, length,width, and radius of CurveCut
        */
        void display() override;
```

```
	/*
	* PROMISES: Copy constructor of CurveCut
	*/
	CurveCut(const CurveCut& r);

	/*
	* PRIMISES: Overlaods assignment operator
	*/
	CurveCut& operator=(CurveCut& rhs);
};
#endif
```

graphicsWorld.h

```
/*
*File Name: Exercise_B, graphicsWorld.h
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/

#ifndef graphics_world
#define graphics_world
class GraphicsWorld {
public :

	//PROMISES: Test multiple inheritance
	void run();
};
#endif // !graphics_world
```

Point.h

```
/*
*File Name: Exercise_B, point.h
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/



#ifndef point_h
#define point_h

class Point {

public:
	/*
	*PROMISES:  static variable to keep track of how many Point objects are created
	*/
	static int counter;

	/*
	* PROMISES: This function returns the distance between two point object
```

```
*/
static double distance(const Point& a, const Point& b);

/*
* PROMISES: This function returns counter
*/
static int get_counter();

/*
* PROMISES: displays the x,y coordinates of this Object
*/
void display() const;

/*
* PROMISES: constructor that sets the x,y private variables
*/
Point(double x, double y);

/*
* PROMISES: return x
*/
double getx() const;

/*
* PROMISES: return y
*/
double gety() const;

/*
* PROMISES: set x
*/
void setx(double x) ;

/*
* PROMISES: set y
*/
void sety(double y);

/*
* PROMISES: return id
*/
int get_id() const;

/*
* PROMISES: overloading assignment operator
*/
Point& operator=(const Point& rhs);

/*
* PROMISES: copy constructor
*/
Point(const Point& r);

/*
* PROMISES: destructor
*/
~Point();
```

```
	/*
	* PROMISES: returns the distance between this and another Point object
	*/
	double distance(const Point& a);

private:
	double x;
	double y;
	int id;

};

#endif
```

Rectangle.h

```
/*
*File Name: Exercise_B,rectangle.h
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/



#include "square.h"
#ifndef rectangle_h
#define rectangle_h
class Rectangle : public Square {
public:
	/*
	* PROMISES: Constructor for Rectangle invokes Shape and Square Constructors
	*/
	Rectangle(double x, double y, double side_a, double side_b, const char* name);

	/*
	* Overides function in Shpae class
	* PROMISES: returns area of  Rectangle
	*/
	double area() const override;

	/*
	* PROMISES: returns side b
	*/
	double get_side_b() const;

	/*
	* PROMISES: sets side b
	*/
	void set_side_b(double num);
	/*
	* Overrides function in Shape class
	* PROMISES: returns perimeter of Rectangle
	*/
	double perimeter() const override;
	/*
	* Overrides funstion in Shape class
	* PROMISES: displays name, coordinates,sides ,area ,and perimeter of Rectangle
	*/
	void display() override;
	/*
	* PROMISES: Copy constructor
	*/
	Rectangle(const Rectangle& r);

	/*
	* PROMISES: overloads assignment operator
	*/
	Rectangle& operator=( Rectangle& rhs);
private:
	double side_b;
};
#endif // !rectangle_h
```

Shape.h

```
/*
*File Name: Exercise_B, shape.h
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/

#include "Point.h"
#include<iostream>
using namespace std;
#ifndef shape_h
#define shape_h
class Shape {
public:
	/*
	* PROMISES: Constructor of Shape that invokes Point constructor
	*/
	Shape(double x, double y,const char* name);
	/*
	* PROMISES: returns the distance between two shapes
	*/
	static double distance(Shape& the_shape, Shape& other);

	/*
	* PROMISES: returns distance between this and another Shape
	*/
	double distance(Shape& other);

	/*
	* PROMISES: deletes shapeNAme in heap
	*/
	virtual~Shape();
	/*
	* PROMISES: copy constructor of Shape
```

```
        */
        Shape(const Shape& r);
        /*
        * PROMISES: overloading assignment oerator
        */
        Shape& operator=(const Shape& rhs);

        /*
        * PROMISES: return counter of Point object
        */
        int get_counter() const;

        /*
        * PROMISES: return Id of Point
        */
        int get_id() const;

        /*
        * Abstact function
        */
        virtual void display() =0;
        /*
        * PROMISES: return name
        */
        const char* getName() const;

        /*
        * Abstact function
        */
        virtual double area() const = 0;

        /*
        * Abstact function
        */
        virtual double perimeter() const = 0;
protected:
        /*
        * PROMISES: return origin
        */
        const Point& getOrigin();
        /*
        * PROMISES: moves the x,y coordinates by dx,dy
        */
        void move(double dx, double dy);

private:
        Point origin;
        char* shapeName;
};

#endif // !shape.h
```

Square.h

```
/*
*File Name: Exercise_B, square.h
```

```
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/

#include "shape.h"
#include<iostream>
using namespace std;
#ifndef square_h
#define square_h
class Square: public virtual Shape{

public:
	/*
	* PROMISES: constructor of Square that invokes Shape constructor
	*/
	Square(double x, double y, double side, const char* name);
	/*
	* Overrides abstract method in Shape
	* PROMISES: return area of square
	*/
	double area() const override;
	/*
	* Overrides abstract method in Shape
	* PROMISES: return perimeter of square
	*/
	double perimeter() const override;
	/*
	* Overrides abstract method in Shape
	* PROMISES: displays name,coordinates,side,area,and perimeter of Square
	*/
	void display() override;

	/*
	* PROMISES: return side of square
	*/
	double get_side_a() const;
	/*
	* PROMISES: sets side of square
	*/
	void set_side_a(double num);
	/*
	* PROMISES: copy constructor of Square
	*/
	Square(const Square& r);
	/*
	* PROMISES: Overloads assignment operator of Square
	*/
	Square& operator=(const Square& rhs);

private:
	double side;

};
#endif // !square_h
```

ii) Source file:

App.cpp

```
/*
*File Name: Exercise_B,app.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/
#include "graphicsWorld.h"
int main() {
	GraphicsWorld test;
	test.run();
}
```

Circle.cpp

```
/*
*File Name: Exercise_B,circle.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/

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

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

double Circle::area() const
{
	return PI * pow(this->radius, 2);
}

double Circle::get_radius() const
{
	return radius;
}

void Circle::set_radius(double num)
{
	radius = num;
}

double Circle::perimeter() const
{
	return 2*PI*radius;
}

void Circle::display()
{
	cout << "\nShape name: " << getName() << endl;
	getOrigin().display();
	cout << "Radius: " << setprecision(4) << radius << endl;
	cout << "Area: " << setprecision(4) << area() << endl;
	cout << "Perimeter: " << setprecision(4) << perimeter() << endl;
}

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

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

curveCut.cpp

```
/*
*File Name: Exercise_B,curveCut.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/

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

CurveCut::CurveCut(double x, double y, double side_a, double side_b, double radius, const
char* name):Shape(x, y, name),Rectangle(x,y,side_a,side_b,name),Circle(x,y,radius,name)
{
	if (radius <= min(side_a, side_b)) {

	}
	else {
		cout <<"\nError, radius didnt meet criteria \n Terminating program..." <<
endl;
		exit(1);
	}
}

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

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

void CurveCut::display()
{
	cout << "\nShape name: " << getName() << endl;
	getOrigin().display();
	cout << "side a: " << setprecision(4) << get_side_a() << endl;
	cout << "side b: " << setprecision(4) << get_side_b() << endl;
	cout << "Radius of cut: " << setprecision(4) << get_radius()<< endl;

}

CurveCut::CurveCut(const CurveCut& r):Shape(r),Rectangle(r),Circle(r)
{
}

CurveCut& CurveCut::operator=(CurveCut& rhs)
{
	if (this != &rhs) {
		Shape::operator=(rhs);
		Rectangle::operator=(rhs);
		Circle::operator=(rhs);


	}
	return *this;
}
```

graphicsWorld.cpp

```
/*
*File Name: Exercise_B,graphicsWorld.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/

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

void GraphicsWorld::run() {
	//Exercise B–Test
cout << "Tessting multiple inheritance...completed by Ziad Chemali" << endl;
#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:"<<endl;
       Rectangle a(5, 7, 12, 15, "RECTANGLE A");
       a.display();

       Rectangle b(16, 7, 8, 9, "RECTANGLE B");

       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 0 // 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();
#endif
#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;

	CurveCut rc(6, 5, 10, 12, 9, "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
}
```

Point.cpp

```
/*
*File Name: Exercise_B,point.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/

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

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

double Point::distance(const Point& a)
{
	return sqrt(pow((x - a.x), 2) + pow((y - a.y), 2));
}

double Point::distance(const Point& a, const Point& b)
{
	return sqrt(pow((b.x-a.x),2)+ pow((b.y - a.y), 2));
}

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

void Point::display() const
{
	cout << "X-Coordinate: " << setprecision(4) << this->x<<endl;
	cout << "Y-Coordinate: " << setprecision(4) << this->y << endl;

}

Point::Point(double x=0, double y=0)
{
	this->x = x;
	this->y = y;
	counter++;
	id = 1001 +counter;
}

double Point::getx() const
{
	return x;
}

double Point::gety() const
{
	return y;
}

void Point::setx(double x)
{
	this->x = x;
}

void Point::sety(double y)
{
	this->y = y;

}

int Point::get_id() const
{
	return id;
}

Point& Point::operator=(const Point& rhs)
{
	if (this != &rhs) {
		this->x = rhs.getx();
		this->y = rhs.gety();
		this->id = rhs.id;

	}
	return *this;
}

Point::Point(const Point& r)
{
	this->x = r.getx();
	this->y = r.gety();
	this->id = r.id;
}


int Point::counter = 0;
```

rectangle.cpp

```
/*
*File Name: Exercise_B,rectangle.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/

#include "rectangle.h"
#include "shape.h"
#include<iomanip>
double Rectangle::area() const
{
	return side_b*get_side_a();
}

double Rectangle::perimeter() const
{
	return side_b*2+get_side_a()*2;
}

void Rectangle::display()
{
	cout << "\nShape name: " << getName() << endl;
	getOrigin().display();
	cout << "side a: " <<setprecision(4)<<get_side_a()<< endl;
	cout << "side b: "<<setprecision(4) << get_side_b() << endl;
	cout << "Area: " << setprecision(4)<<area()<< endl;
	cout << "Perimeter: " << setprecision(4) << perimeter() << endl;
}

Rectangle::Rectangle(const Rectangle& r):Shape(r),Square(r)
{
	side_b = r.side_b;

}


Rectangle& Rectangle::operator=( Rectangle& rhs)
{

	if (this != &rhs) {
		Shape::operator=(rhs);
		Square::operator=(rhs);
		side_b = rhs.side_b;

	}
	return *this;
}

double Rectangle::get_side_b() const
{
	return side_b;
}

void Rectangle::set_side_b(double num)
{
	this->side_b = num;
}

Rectangle::Rectangle(double x, double y, double side_a, double side_b, const char* name)
:Shape(x,y,name),Square(x, y, side_a, name) {
	this->side_b = side_b;
}
```

Shape.cpp

```
/*
*File Name: Exercise_B,shape.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/

#include "shape.h"
#include "Point.h"
#include <string>
#include <cassert>

Shape::Shape(double x, double y, const char* name) :origin(x, y)
{
	const char* temp = name;
	int n = 0;
	while (*temp) {
		n++;
		temp++;
	}
	if (n > 0) {

	this->shapeName = new char[n+1];//to include \0

	for (int i = 0;i < n;i++) {
		shapeName[i] = name[i];
	}
	shapeName[n] = '\0';
}
	else {
		cout << "Name parameter is empty" << endl;
	}
}

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

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

Shape::Shape(const Shape& r):origin(r.origin)
{

	delete[] shapeName;
	const char* temp = r.getName();
	int n = 0;
	while (*temp) {
		n++;
		temp++;
	}
	if (n > 0) {

		this->shapeName = new char[n + 1];//to include \0

		for (int i = 0;i < n;i++) {
			shapeName[i] = r.getName()[i];
		}
		shapeName[n] = '\0';

	}

}

Shape& Shape::operator=(const Shape& rhs)
{
	if (this!= &rhs) {
		origin = rhs.origin;
		delete[] shapeName;
		const char* temp = rhs.getName();
```

```
			int n = 0;
			while (*temp) {
				n++;
				temp++;
			}
			if (n > 0) {

				this->shapeName = new char[n + 1];//to include \0

				for (int i = 0;i < n;i++) {
					shapeName[i] = rhs.getName()[i];
				}
				shapeName[n] = '\0';

			}

		}
		return *this;
}

int Shape::get_counter() const
{
		return this->origin.get_counter();
}

int Shape::get_id() const
{
		return origin.get_id();
}

const Point& Shape::getOrigin()
{
		return  origin;// TODO: insert return statement here
}

const char* Shape::getName() const
{
		return shapeName;
}




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

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

Square.cpp

```
/*
*File Name: Exercise_B,square.cpp
* Lab_5
* Completed by Ziad Chemali
* Submission: 23,10,2020
*/

#include "square.h"
#include <iomanip>
Square::Square(double x, double y, double side, const char* name):Shape(x, y, name)
{
	this->side = side;

}

double Square::get_side_a() const
{
	return side;
}

void Square::set_side_a(double num)
{
	side = num;
}

Square::Square(const Square& r):Shape(r)
{

	this->set_side_a(r.get_side_a());
}

Square& Square::operator=(const Square& rhs)
{
	if (this != &rhs) {
		Shape::operator=(rhs);
		this->side=rhs.get_side_a();

	}
	return *this;
}

double Square::area() const {

		return pow(side, 2);
	}


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

void Square::display()
{
	cout << "\nShape name: " << getName() << endl;
	getOrigin().display();
	cout << "side a: " << setprecision(4) << side << endl;
	cout << "Area: " <<setprecision(4) <<area()<< endl;
	cout << "Perimeter: " << setprecision(4) << perimeter() << endl;
}
```

2) Code Output:

```
Testing multiple inheritance...completed by Ziad Chemali


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:


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


Distance between square 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 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.



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: 2e+04
Perimeter: 600

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.



Shape name: RECTANGLE A
X-Coordinate: 5
Y-Coordinate: 7
side a: 100
side b: 200

Area: 2e+04

Perimeter: 600


Testing Functions in class Circle:


Shape name: CIRCLE C

X-Coordinate: 3

Y-Coordinate: 5

Radius: 9

Area: 254.5

Perimeter: 56.55

the area of CIRCLE C is: 254.5

the perimeter of CIRCLE C is: 56.55


The distance between rectangle a and circle c is: 2.828

Shape name: CurveCut rc

X-Coordinate: 6

Y-Coordinate: 5

side a: 10

side b: 12

Radius of cut: 9

the area of CurveCut rc is: 56.38the perimeter of CurveCut rc is: 40.14

The distance between rc and c is: 3

Shape name: SQUARE - S

X-Coordinate: 5

Y-Coordinate: 7

side a: 12

Area: 144

Perimeter: 48
```

```
the area of SQUARE - Sis: 144

the perimeter of SQUARE - S is: 48

Shape name: RECTANGLE A

X-Coordinate: 5

Y-Coordinate: 7

side a: 400

side b: 300

Area: 1.2e+05

Perimeter: 1400


the area of RECTANGLE Ais: 1.2e+05

the perimeter of SQUARE - S is: 1400

Shape name: CIRCLE C

X-Coordinate: 3

Y-Coordinate: 5

Radius: 9

Area: 254.5

Perimeter: 56.55


the area of CIRCLE Cis: 254.5

the circumference of CIRCLE C is: 56.55

Shape name: CurveCut rc

X-Coordinate: 6

Y-Coordinate: 5

side a: 10

side b: 12

Radius of cut: 9


the area of CurveCut rcis: 56.38

the perimeter of CurveCut rc is: 40.14

Testing copy constructor in class CurveCut:
```

```
Shape name: CurveCut rc

X-Coordinate: 6

Y-Coordinate: 5

side a: 10

side b: 12

Radius of cut: 9


Testing assignment operator in class CurveCut:


Shape name: CurveCut cc2

X-Coordinate: 2

Y-Coordinate: 5

side a: 100

side b: 12

Radius of cut: 9


Shape name: CurveCut rc

X-Coordinate: 6

Y-Coordinate: 5

side a: 10

side b: 12

Radius of cut: 9
```