ENSF 614

Advanced System Analysis and Software Design

LAB 5

Author:

Steven Duong
(30022492)

Affiliation
Department of Electrical and Software Engineering
University of Calgary
Calgary, Alberta

Lab Block: B01
Date of Report: Mar 5, 2023

Exercise A – Header File point.h

```
/*
 *  File Name: point.h
 *  Assignment: ENSF 614 Lab 5 Exercise A
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#ifndef LAB_5_POINT_H
#define LAB_5_POINT_H

class Point {

  private: double xCoordinate,
  yCoordinate; // the x and y coordinates of the point
  int id; // the id number of the point
  static int count; // the number of objects of class Point created so
far

  public: Point(double x, double y);
  // REQUIRES
  //     two arguments of type double
  // PROMISES
  //     creates a Point object with arguments a and b
  //     of type double and initializes data members.

  ~Point();
  // PROMISES
  //    destroy the Point object and deallocate the memory
  //    decrement the number of objects by 1

  Point(const Point & P);
  // REQUIRES
  //     P as a reference to a constant Point object
  // PROMISES
  //     creates a deep copy of Point object with the data
  //     members of P.

  Point & operator = (const Point & rhs);
  // REQUIRES
  //     rhs as a reference to a constant Point object
  // PROMISES
  //     deep copy of data members of rhs to the object
  //     being created

  void display() const;
  // PROMISES
```

```
  //     prints the x and y coordinates of the Point object

  double getx() const;
  // PROMISES
  //     returns the x coordinate of Point object

  double gety() const;
  // PROMISES
  //     returns the y coordinate of Point object

  void setx(double x);
  // REQUIRES
  //     the argument to be of type double
  // PROMISES
  //     sets the x coordinate of the Point object to be x

  void sety(double y);
  // REQUIRES
  //     the argument to be of type double
  // PROMISES
  //     sets the y coordinate of the Point object to be y

  int counter() const;
  // PROMISES
  //     returns the num_of_objects of class Point

  double distance(const Point & p) const;
  // REQUIRES
  //     P to be a reference to a constant Point object
  // PROMISES
  //     returns the distance between this point and the given point

  static double distance(const Point & p1,
    const Point & p2);
  // REQUIRES
  //     p1 and p2 to be references to Point objects
  // PROMISES
  //     returns the distance between p1 and p2

};

#endif //LAB_5_POINT_H
```

Exercise A – Source File point.cpp

```
/*
 *  File Name: point.cpp
 *  Assignment: ENSF 614 Lab 5 Exercise A
```

```
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#include "point.h"

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

int Point::count = 1000;

Point::Point(double x, double y) {
  this -> xCoordinate = x;
  this -> yCoordinate = y;
  this -> id = ++count;
}

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

Point::Point(const Point & P) {
  this -> xCoordinate = P.getx();
  this -> yCoordinate = P.gety();
  this -> id = ++count;
}

Point & Point::operator = (const Point & rhs) {
  if (this != & rhs) {
    this -> xCoordinate = rhs.getx();
    this -> yCoordinate = rhs.gety();
    this -> id = ++count;
  }

  return * this;
}

void Point::display() const {
  cout << "X-coordinate: " << fixed << setprecision(2) << getx() <<
endl;
  cout << "Y-coordinate: " << fixed << setprecision(2) << gety() <<
endl;
}
```

```cpp
double Point::getx() const {
  return this -> xCoordinate;
}

double Point::gety() const {
  return this -> yCoordinate;
}

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

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

int Point::counter() const {
  return count;
}

double Point::distance(const Point & p) const {
  double dx = this -> getx() - p.getx();
  double dy = this -> gety() - p.gety();

  return sqrt(pow(dx, 2) + pow(dy, 2));
}

double Point::distance(const Point & p1,
  const Point & p2) {
  double dx = p1.getx() - p2.getx();
  double dy = p1.gety() - p2.gety();

  return sqrt(pow(dx, 2) + pow(dy, 2));
}
```

Exercise A – Header File shape.h

```cpp
/*
 *  File Name: shape.h
 *  Assignment: ENSF 614 Lab 5 Exercise A
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#ifndef LAB_5_SHAPE_H
#define LAB_5_SHAPE_H
```

```
#include "point.h"

class Shape {

  protected: Point origin;
  char * shapeName;

  public: Shape(double x, double y,
    const char * name);
  // REQUIRES
  //     an argument of type Point and a char pointer to its name
  // PROMISES
  //     initialize the data members of Shape

  virtual~Shape();
  // REQUIRES
  //     destroys the shape object and deallocates the memory

  Shape(const Shape & s);
  // REQUIRES
  //     s as a reference to a constant Shape object
  // PROMISES
  //     deep copy of Shape object with data members of s

  Shape & operator = (const Shape & rhs);
  // REQUIRES
  //     rhs as a reference of constant Shape object
  // PROMISES
  //     deep copy of a Shape object with data members of rhs

  const Point & getOrigin() const;
  // PROMISES
  //     returns the reference to the origin of a Point object

  const char * getName() const;
  // PROMISES
  //     returns the pointer to the shape name

  virtual void display() const;
  // PROMISES
  //     display the shape's name, and x and y coordinates of the
Point object

  virtual double distance(Shape & other) const;
  // REQUIRES
  //     other as a reference to a Shape object
  // PROMISES
  //     the distance between this Shape and other on the cartesian
plane
```

```
  static double distance(Shape & the_shape, Shape & other);
  // REQUIRES
  //     the_shape and other as references to Shape objects
  // PROMISES
  //     the distance between the_shape and other on the cartesian
plane

  void move(double dx, double dy);
  // REQUIRES
  //     dx and dy as type double
  // PROMISES
  //     change the position of the Shape by dx and dy

  virtual double area() const = 0; // pure virtual (abstract) method
  // PROMISES
  //     returns the area of the shape object

  virtual double perimeter() const = 0; // pure virtual (abstract)
method
  // PROMISES
  //     returns the perimeter of the shape object
};

#endif //LAB_5_SHAPE_H
```

Exercise A – Source File shape.cpp

```
/*
 *  File Name: shape.cpp
 *  Assignment: ENSF 614 Lab 5 Exercise A
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#include "shape.h"

#include <cstring>

#include <iostream>

#include <iomanip>

using namespace std;

Shape::Shape(double x, double y,
  const char * name): origin(Point(x, y)) {
  this -> shapeName = new char[strlen(name) + 1];
```

```
  strcpy(this -> shapeName, name);
}

Shape::~Shape() {
  delete[] this -> shapeName;
  this -> shapeName = nullptr;
}

Shape::Shape(const Shape & s): origin(Point(s.getOrigin().getx(),
s.getOrigin().gety())) {
  this -> shapeName = new char[strlen(s.getName()) + 1];
  strcpy(this -> shapeName, s.getName());
}

Shape & Shape::operator = (const Shape & rhs) {
  if (this != & rhs) {
    delete[] this -> shapeName;
    this -> origin = Point(rhs.getOrigin().getx(),
rhs.getOrigin().gety());
    this -> shapeName = new char[strlen(rhs.getName()) + 1];
    strcpy(this -> shapeName, rhs.getName());
  }

  return * this; //deference since this is pointer
}

const Point & Shape::getOrigin() const {
  return this -> origin;
}

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

void Shape::display() const {
  cout << "Shape Name: " << shapeName << '\n';
  cout << "X-coordinate: " << fixed << setprecision(2) <<
origin.getx() << '\n';
  cout << "Y-coordinate: " << fixed << setprecision(2) <<
origin.gety() << '\n';
}

double Shape::distance(Shape & other) const {
  return this -> getOrigin().distance(other.getOrigin());
}

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

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

Exercise A – Header File square.h

```
/*
 *  File Name: square.h
 *  Assignment: ENSF 614 Lab 5 Exercise A
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#ifndef LAB_5_SQUARE_H
#define LAB_5_SQUARE_H

#include "point.h"

#include "shape.h"

class Square: virtual public Shape {
  protected: double side_a;

  public: Square(double x, double y, double side,
    const char * name);
  // REQUIRES
  //     three arguments of type double and a pointer to a string
literal
  // PROMISES
  //     create a square object with the given arguments

  double area() const override;
  // PROMISES
  //     returns the area of a square

  double perimeter() const override;
  // PROMISES
  //     returns the perimeter of a square

  double get_side_a() const;
  // PROMISES
  //     returns the side of a square

  void set_side_a(double side);
  // REQUIRES
  //     side argument of type double
```

```
  // PROMISES
  //     sets the side of square

  void display() const override;
  // PROMISES
  //     prints the square object
};

#endif //LAB_5_SQUARE_H
```

Exercise A – Source File square.cpp

```
/*
 *  File Name: square.cpp
 *  Assignment: ENSF 614 Lab 5 Exercise A
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#include "square.h"

#include <iostream>

#include <iomanip>

using namespace std;

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

double Square::area() const {
  return this -> side_a * this -> side_a;
}

double Square::perimeter() const {
  return 4 * this -> side_a;
}

double Square::get_side_a() const {
  return this -> side_a;
}

void Square::set_side_a(double side) {
  this -> side_a = side;
}
```

```
void Square::display() const {
  cout << "Square Name: " << getName() << endl;
  getOrigin().display();
  cout << "Side a: " << fixed << setprecision(2) << get_side_a() <<
endl;
  cout << "Area: " << fixed << setprecision(2) << area() << endl;
  cout << "Perimeter: " << fixed << setprecision(2) << perimeter() <<
endl;
}
```

Exercise A – Header File rectangle.h

```
/*
 *  File Name: rectangle.h
 *  Assignment: ENSF 614 Lab 5 Exercise A
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#ifndef LAB_5_RECTANGLE_H
#define LAB_5_RECTANGLE_H

#include "square.h"

class Rectangle: public Square {

  protected: double side_b;

  public: Rectangle(double x, double y, double side_a, double side_b,
    const char * name);
  // REQUIRES
  //     x, y, side_a and side_b as type double
  //     name as string static c-string
  // PROMISES
  //     create a rectangle object with the arguments given

  double area() const override;
  // PROMISES
  //     returns the area of a rectangle

  double perimeter() const override;
  // PROMISES
  //     returns the perimeter of a rectangle

  double get_side_b() const;
  // PROMISES
```

```
  //     returns side_b of the rectangle

  void set_side_b(double side_b);
  // REQUIRES
  //     side_b to be of type double
  // PROMISES
  //     sets side_b to be the value given by the argument

  void display() const override;
  // PROMISES
  //     prints the Rectangle object
};

#endif //LAB_5_RECTANGLE_H
```

Exercise A – Source File rectangle.cpp

```
/*
 *  File Name: rectangle.cpp
 *  Assignment: ENSF 614 Lab 5 Exercise A
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#include "rectangle.h"

#include <iostream>

#include <iomanip>

using namespace std;

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;
}

double Rectangle::area() const {
  return this -> side_a * this -> side_b;
}

double Rectangle::perimeter() const {
  return 2 * this -> side_a + 2 * this -> side_b;
}

double Rectangle::get_side_b() const {
  return this -> side_b;
```

```
}

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

void Rectangle::display() const {
  cout << "Rectangle Name: " << getName() << endl;
  getOrigin().display();
  cout << "Side a: " << fixed << setprecision(2) << get_side_a() <<
endl;
  cout << "Side b: " << fixed << setprecision(2) << get_side_b() <<
endl;
  cout << "Area: " << fixed << setprecision(2) << area() << endl;
  cout << "Perimeter: " << fixed << setprecision(2) << perimeter() <<
endl;
}
```

Exercise A – Header File graphicsWorld.h

```
/*
 *  File Name: graphicsWorld.h
 *  Assignment: ENSF 614 Lab 5 Exercise A, B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#ifndef LAB_5_GRAPHICSWORLD_H
#define LAB_5_GRAPHICSWORLD_H

class GraphicsWorld {
  public: static void run();
  // PROMISES
  //     to test the functionalities of class point, shape, square and
rectangle
};

#endif //LAB_5_GRAPHICSWORLD_H
```

Exercise A – Source File graphicsWorld.cpp

```
/*
 *  File Name: graphicsWorld.cpp
 *  Assignment: ENSF 614 Lab 5 Exercise A, B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
```

```
 *  Submission Date: Mar 5, 2023
 */

#include "graphicsWorld.h"

#include "point.h"

#include "shape.h"

#include "square.h"

#include "circle.h"

#include "rectangle.h"

#include "curvecut.h"

#include <iostream>

using namespace std;

void GraphicsWorld::run() {
  // #if 0 // Change 0 to 1 to test Point
  Point m(6, 8);
  Point n(6, 8);
  n.setx(9);
  cout << "Author: Steven Duong" << endl;
  cout << "\nExercise A" << endl;
  cout << "----------" << endl;
  cout << "\nExpected to display 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 0 // 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 0 // Change 0 to 1 to test Rectangle
  cout << "\nTesting Functions in class Rectangle:\n";
  Rectangle a(5, 7, 12, 15, "RECTANGLE A");
  a.display();
  Rectangle b(16, 7, 8, 9, "RECTANGLE B");
  b.display();
  double d = a.distance(b);
  cout << "\nDistance between square a, and b is: " << d << endl;
  Rectangle rec1 = a;
```

```
  rec1.display();
  cout << "\nTesting assignment operator in class Rectangle:" << endl;
  Rectangle rec2(3, 4, 11, 7, "RECTANGLE rec2");
  rec2.display();
  rec2 = a;
  a.set_side_b(200);
  a.set_side_a(100);
  cout << "\nExpected to display the following values for 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 << "\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 object 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 // end of block to test array of pointer and polymorphism

  /**********************ASSUME THIS CODE SEGMENT FOR EXERCISE A IS HERE
   *********************/
  //#if 0
```

```
  cout << "\nExercise B" << endl;
  cout << "----------" << endl;
  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 << "The distance between rectangle a and circle c is: " << d <<
endl << endl;
  CurveCut rc(6, 5, 10, 12, 9, "CurveCut rc");
  rc.display();
  cout << "the area of " << rc.getName() << " is: " << rc.area() <<
endl;
  cout << "the perimeter of " << rc.getName() << " is: " <<
rc.perimeter();
  d = rc.distance(c);
  cout << "\nThe distance between rc and c is: " << d << endl;
  // Using an array of Shape pointers:
  // Shape* sh[4];
  cout << endl;
  sh[0] = & s;
  sh[1] = & a;
  sh[2] = & c;
  sh[3] = & rc;
  sh[0] -> display();
  cout << "the area of " << sh[0] -> getName() << " is: " << sh[0] ->
area() << endl;
  cout << "the perimeter of " << sh[0] -> getName() << " is: " <<
sh[0] -> perimeter() << endl << endl;
  sh[1] -> display();
  cout << "the area of " << sh[1] -> getName() << " is: " << sh[1] ->
area();
  cout << "\nthe perimeter of " << sh[1] -> getName() << " is: " <<
sh[1] -> perimeter() << endl << endl;
  sh[2] -> display();
  cout << "the area of " << sh[2] -> getName() << " is: " << sh[2] ->
area();
  cout << "\nthe circumference of " << sh[2] -> getName() << " is: "
<< sh[2] -> perimeter() << endl << endl;
  sh[3] -> display();
  cout << "the area of " << sh[3] -> getName() << " is: " << sh[3] ->
area();
  cout << "\nthe perimeter of " << sh[3] -> getName() << " is: " <<
sh[3] -> perimeter() << endl;
  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;
  cout << endl;
  cc2.display();
}
```

Exercise A – Source File lab5exeA.cpp

```
/*
 *  File Name: lab5exeA
 *  Assignment: ENSF 614 Lab 5 Exercise A
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#include "graphicsWorld.h"

using namespace std;

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

Exercise A – Program Output

```
"/Users/stevenduong/CLionProjects/ENSF 614/Labs/Lab 5/cmake-build-debug/Lab_5"
Author: Steven Duong

Exercise A
----------

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

```
Testing Functions in class Rectangle:
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 12.00
Side b: 15.00
Area: 180.00
Perimeter: 54.00
Rectangle Name: RECTANGLE B
X-coordinate: 16.00
Y-coordinate: 7.00
Side a: 8.00
Side b: 9.00
Area: 72.00
Perimeter: 34.00

Distance between square a, and b is: 11.00
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 12.00
Side b: 15.00
Area: 180.00
Perimeter: 54.00

Testing assignment operator in class Rectangle:
Rectangle Name: RECTANGLE rec2
X-coordinate: 3.00
Y-coordinate: 4.00
Side a: 11.00
Side b: 7.00
Area: 77.00
Perimeter: 36.00
```

```
Expected to display the following values for 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.

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

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

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

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

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

Testing array of pointers and polymorphism:
Square Name: SQUARE - S
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 12.00
Area: 144.00
Perimeter: 48.00
Rectangle Name: RECTANGLE B
X-coordinate: 16.00
Y-coordinate: 7.00
Side a: 8.00
Side b: 9.00
Area: 72.00
Perimeter: 34.00
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 12.00
Side b: 15.00
Area: 180.00
Perimeter: 54.00
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 100.00
Side b: 200.00
Area: 20000.00
Perimeter: 600.00

Process finished with exit code 0
```

Exercise B – Header File circle.h

```
/*
 *  File Name: circle.h
 *  Assignment: ENSF 614 Lab 5 Exercise B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#ifndef LAB_5_CIRCLE_H
#define LAB_5_CIRCLE_H

#include "shape.h"

class Circle: virtual public Shape {
  protected: double radius;

  public: Circle(double x, double y, double r,
    const char * name);
  // REQUIRES
  //     x, y and r as type double
  //     name as static c-string
  // PROMISES
  //     create a circle object with the arguments given

  double area() const override;
  // PROMISES
  //     returns the area of the circle

  double perimeter() const override;
  // PROMISES
  //     returns the perimeter of the circle

  double get_r() const;
  // PROMISES
  //     returns the radius of the circle

  void set_r(double r);
  // REQUIRES
  //     r as type double
  // PROMISES
  //     sets the radius of the circle to the value of r

  void display() const override;
  // PROMISES
  //     prints the circle object
};
```

```
#endif //LAB_5_CIRCLE_H
```

Exercise B – Source File circle.cpp

```
/*
 *  File Name: circle.cpp
 *  Assignment: ENSF 614 Lab 5 Exercise B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#include "circle.h"

#include <cmath>

#include <iostream>

#include <iomanip>

using namespace std;

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

double Circle::area() const {
  return M_PI * this -> radius * this -> radius;
}

double Circle::perimeter() const {
  return 2 * M_PI * this -> radius;
}

double Circle::get_r() const {
  return this -> radius;
}

void Circle::set_r(double r) {
  this -> radius = r;
}

void Circle::display() const {
  cout << "Circle Name: " << getName() << endl;
  getOrigin().display();
  cout << "Radius: " << fixed << setprecision(2) << get_r() << endl;
  cout << "Area: " << fixed << setprecision(2) << area() << endl;
```

```
  cout << "Perimeter: " << fixed << setprecision(2) << perimeter() <<
endl;

}
```

Exercise B – Header File curvecut.h

```
/*
 *  File Name: curvecut.h
 *  Assignment: ENSF 614 Lab 5 Exercise B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#ifndef LAB_5_CURVECUT_H
#define LAB_5_CURVECUT_H

#include "rectangle.h"

#include "circle.h"


class CurveCut: public Rectangle, public Circle {

  public: CurveCut(double x, double y, double side_a, double side_b,
double r,
    const char * name);
  // REQUIRES
  //     x, y, side_a, side_b, r to be of type double
  //     name as a static c-string
  // PROMISES
  //     create a CurveCut object with the given args

  double area() const;
  // PROMISES
  //     calculate the area of the CurveCut object

  double perimeter() const;
  // PROMISES
  //     calculate the perimeter of the CurveCut object

  void display() const;
  // PROMISES
  //     print the CurveCut object
};

#endif //LAB_5_CURVECUT_H
```

Exercise B – Source File curvecut.cpp

```
/*
 *  File Name: curvecut.cpp
 *  Assignment: ENSF 614 Lab 5 Exercise B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 5, 2023
 */

#include "curvecut.h"

#include <iostream>

using namespace std;

CurveCut::CurveCut(double x, double y, double side_a, double side_b,
double r,
    const char * name): Shape(x, y, name), Rectangle(x, y, side_a,
side_b, name),
  Circle(x, y, r, name) {
    if (!(r <= side_a && r <= side_b)) {
      cerr << "The radius must be less than or equal to the width and
length.";
      exit(1);
    }
  }

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

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

void CurveCut::display() const {
  cout << "CurveCut Name: " << this -> getName() << endl;
  Circle::getOrigin().display();
  cout << "Width: " << this -> get_side_a() << endl;
  cout << "Length: " << this -> get_side_b() << endl;
  cout << "Radius of the cut: " << this -> get_r() << endl;
}
```

Exercise B – Program Output

```
Exercise B
----------

Testing Functions in class Circle:
Circle Name: CIRCLE C
X-coordinate: 3.00
Y-coordinate: 5.00
Radius: 9.00
Area: 254.47
Perimeter: 56.55
the area of CIRCLE C is: 254.47
the perimeter of CIRCLE C is: 56.55
The distance between rectangle a and circle c is: 2.83

CurveCut Name: CurveCut rc
X-coordinate: 6.00
Y-coordinate: 5.00
Width: 10.00
Length: 12.00
Radius of the cut: 9.00
the area of CurveCut rc is: 56.38
the perimeter of CurveCut rc is: 40.14
The distance between rc and c is: 3.00

Square Name: SQUARE - S
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 12.00
Area: 144.00
Perimeter: 48.00
the area of SQUARE - S is: 144.00
the perimeter of SQUARE - S is: 48.00
```

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

Circle Name: CIRCLE C
X-coordinate: 3.00
Y-coordinate: 5.00
Radius: 9.00
Area: 254.47
Perimeter: 56.55
the area of CIRCLE C is: 254.47
the circumference of CIRCLE C is: 56.55

CurveCut Name: CurveCut rc
X-coordinate: 6.00
Y-coordinate: 5.00
Width: 10.00
Length: 12.00
Radius of the cut: 9.00
the area of CurveCut rc is: 56.38
the perimeter of CurveCut rc is: 40.14

Testing copy constructor in class CurveCut:
CurveCut Name: CurveCut rc
X-coordinate: 6.00
Y-coordinate: 5.00
Width: 10.00
Length: 12.00
Radius of the cut: 9.00
```

```
Testing assignment operator in class CurveCut:
CurveCut Name: CurveCut cc2
X-coordinate: 2.00
Y-coordinate: 5.00
Width: 100.00
Length: 12.00
Radius of the cut: 9.00

CurveCut Name: CurveCut rc
X-coordinate: 6.00
Y-coordinate: 5.00
Width: 10.00
Length: 12.00
Radius of the cut: 9.00

Process finished with exit code 0
```