**Name:** Antonio Santos, Christina Lu

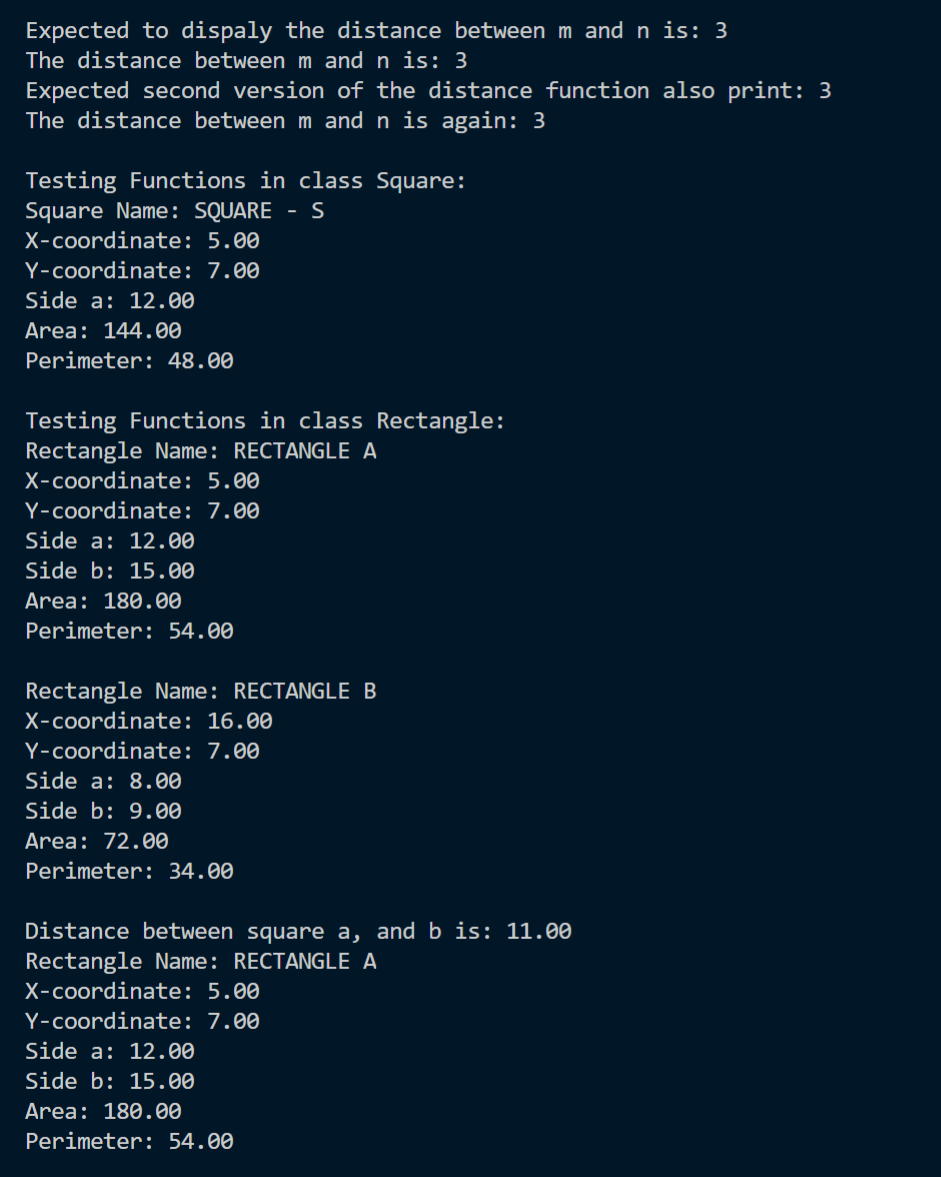
**Course Name:** Principles of Software Design

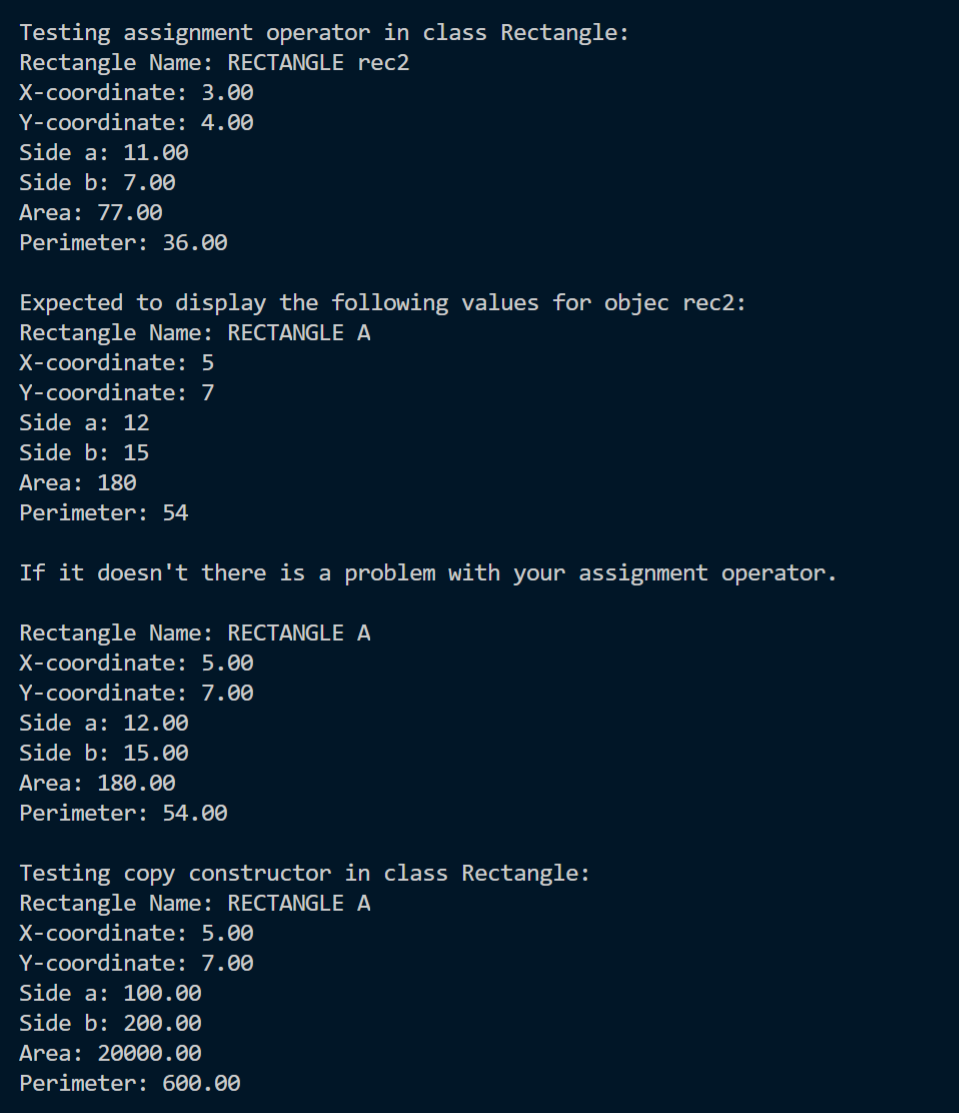
**Course Code:** ENSF 480

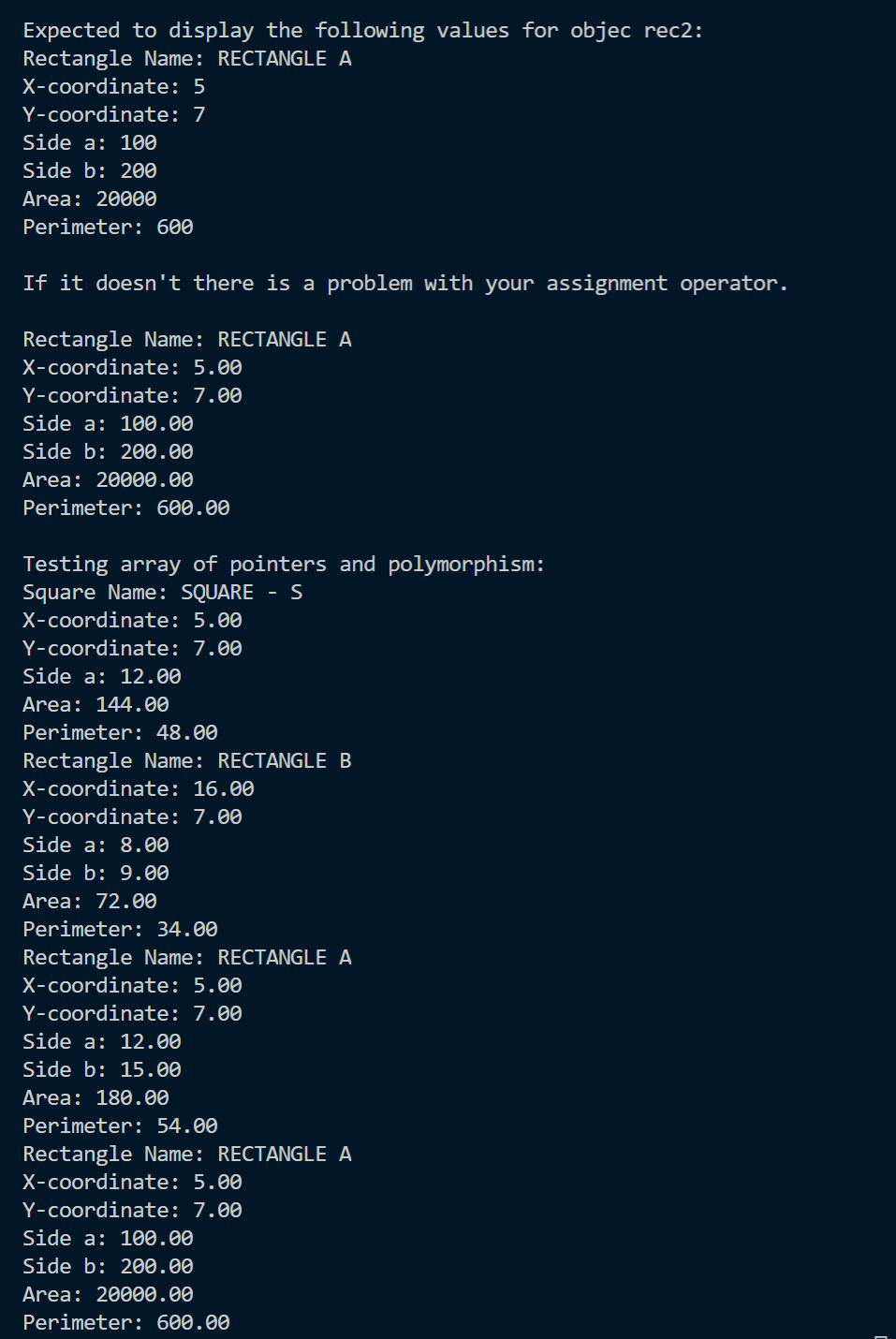
**Assignment Number:** Lab 2

**Submission Date:** Oct 4, 2019

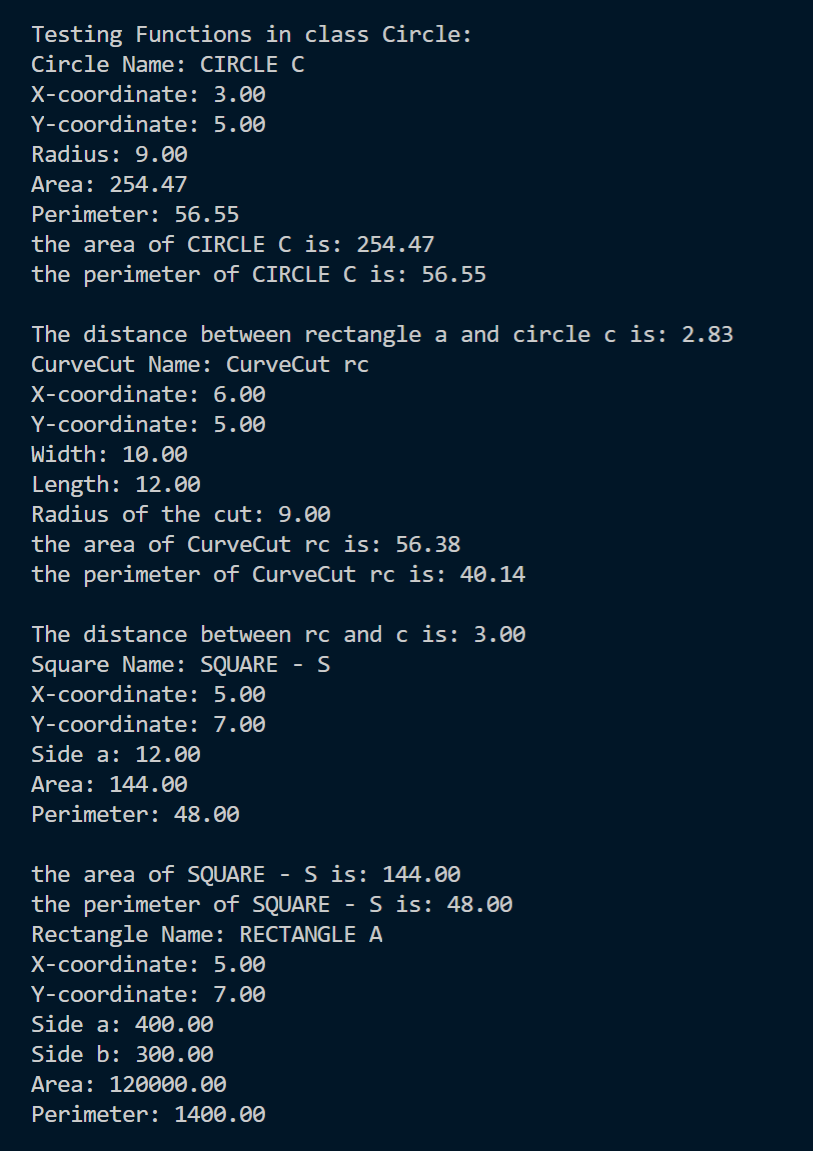
Exercise A Outputs

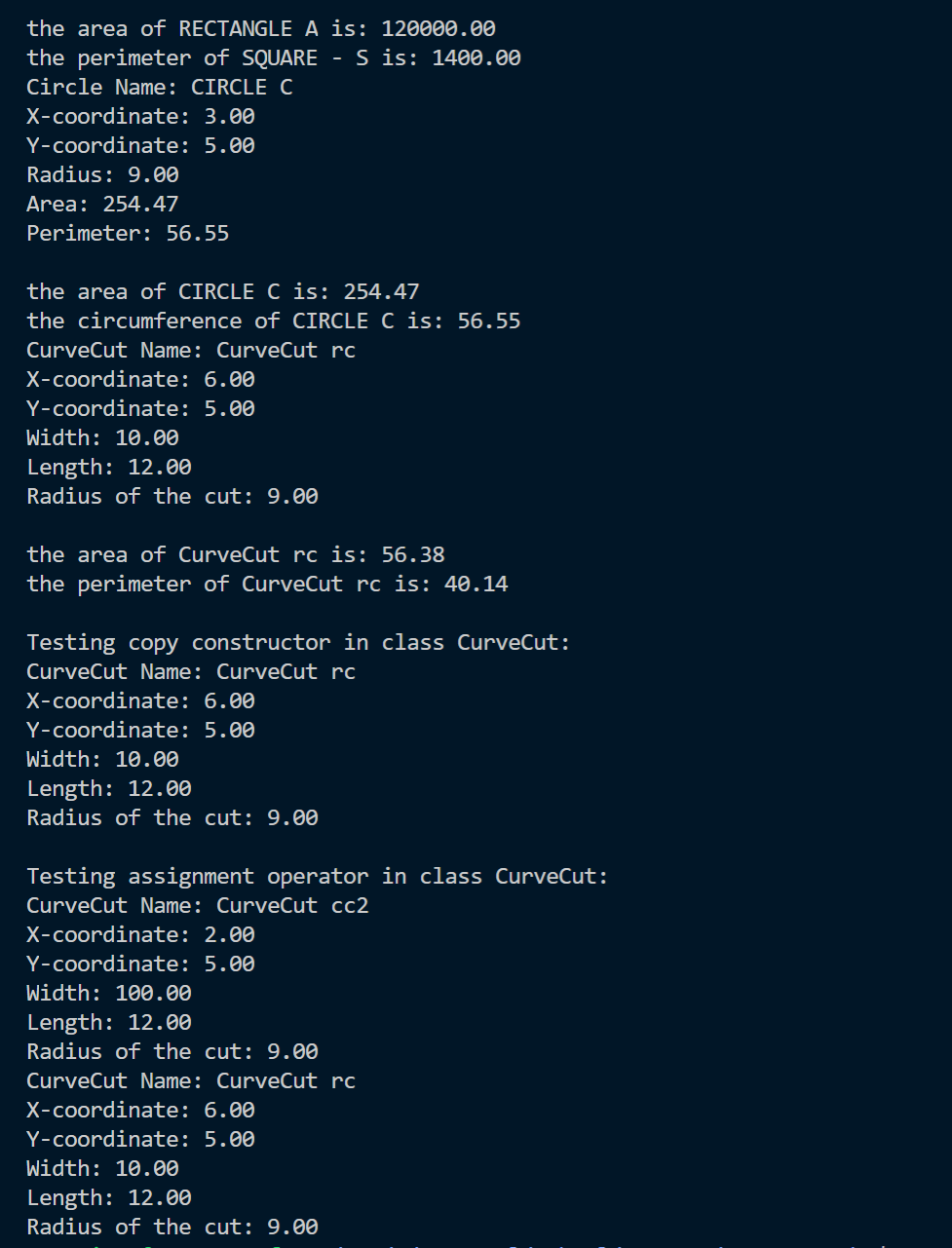






Exercise B Output





ExA Code Segment

ExB Code Segment

//*main.cpp*

#*include* "graphicsWorld.h"

using namespace std;

int *main*(){

  GraphicsWorld::*run*();

*return* 0;

}

//*point.cpp*

#*include* <stdlib.h>

#*include* <stdio.h>

#*include* <iostream>

#*include* <cmath>

#*include* "point.h"

using namespace std;

int Point::count = 0;

Point::*Point*(double x, double y): *x*(x), *y*(y), *id*(count + 1000) { count++; }

double Point::*get\_x*() *const* { *return* x; }

double Point::*get\_y*() *const* { *return* y; }

int Point::*get\_id*() *const* { *return* id; }

void Point::*set\_x*(double val) { x = val; }

void Point::*set\_y*(double val) { y = val; }

void Point::*set\_id*(int val) { id = val; }

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

void Point::*display*(){

  cout<<std::fixed;

  cout.*precision*(2);

  cout<<"X-coordinate: "<<*get\_x*()<<endl;

  cout<<"Y-coordinate: "<<*get\_y*()<<endl;

};

double Point::*distance*(*const* Point& other) *const* {

  double distance;

  double xDist = other.x - x;

  double yDist = other.y - y;

  distance = *sqrt*(*pow*(xDist, 2) + *pow*(yDist, 2));

*return* distance;

};

double Point::*distance*(*const* Point& p1, *const* Point& p2) {

  double distance;

  double xDist = p1.x - p2.x;

  double yDist = p1.y - p2.y;

  distance = *sqrt*(*pow*(xDist, 2) + *pow*(yDist, 2));

*return* distance;

};

//*point.h*

#*ifndef* *POINT\_H*

#*define* *POINT\_H*

class Point{

  public:

*Point*(double x, double y);

    void *display*();

*static* int *counter*();

    double *distance*(*const* Point& other) *const*;

*static* double *distance*(*const* Point& p1, *const* Point& p2);

    double *get\_x*() *const*;

    double *get\_y*() *const*;

    int *get\_id*() *const*;

    void *set\_x*(double val);

    void *set\_y*(double val);

    void *set\_id*(int val);

  private:

    double x;

    double y;

    int id;

*static* int count;

};

#*endif*

//*shape.cpp*

#*include* <stdlib.h>

#*include* <stdio.h>

#*include* <iostream>

#*include* <cmath>

#*include* <string.h>

#*include* "shape.h"

using namespace std;

Shape::*Shape*(double x, double y, *const* char\* name): *origin*(*Point*(x,y)) {

  shapeName = new char[(int)*strlen*(name)];

*strcpy*(shapeName, name);

};

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

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

void Shape::*display*() *const* {

  cout<<"Shape Name: "<<*getName*()<<endl;

  Point o = *getOrigin*();

  o.*display*();

}

double Shape::*distance*(Shape& other) *const*{

*return* origin.*distance*(other.origin);

}

double Shape::*distance*(Shape& the\_shape, Shape& other){

  double distance;

  double xDist = the\_shape.*getOrigin*().*get\_x*() - other.*getOrigin*().*get\_x*();

  double yDist = the\_shape.*getOrigin*().*get\_y*() - other.*getOrigin*().*get\_y*();

  distance = *sqrt*(*pow*(xDist, 2) + *pow*(yDist, 2));

*return* distance;

}

void Shape::*move*(double dx, double dy){

  double newx = origin.*get\_x*() + dx;

  double newy = origin.*get\_y*() + dy;

  origin.*set\_x*(newx);

  origin.*set\_y*(newy);

}

Shape::*~Shape*(){

delete [] shapeName;

}

Shape::*Shape*(*const* Shape& source):*origin*(source.origin), *shapeName*(new

char[*strlen*(source.shapeName)+1]){

*if*(shapeName == NULL){

    cerr << "Memory not available...";

*exit*(1);

  }

*strcpy*(shapeName, source.shapeName);

}

Shape& Shape::operator=(*const* Shape& rhs){

*if*(this==&rhs)

*return* \*this;

  delete [] shapeName;

  shapeName = new char[*strlen*(rhs.shapeName)+1];

*if*(shapeName == NULL){

    cerr << "Memory not available...";

*exit*(1);

    }

*strcpy*(shapeName, rhs.shapeName);

  origin = rhs.origin;

*return* \*this;

}

double Shape::*area*(){

*return* 0;

}

double Shape::*perimeter*(){

*return* 0;

}

void Shape::*display*(){

*return*;

}

//*shape.h*

#*ifndef* *SHAPE\_H*

#*define* *SHAPE\_H*

#*include* "point.h"

class Shape{

  public:

*Shape*(double x, double y, *const* char\* shapeName);

*virtual* *~Shape*();

    Shape& operator=(*const* Shape& rhs);

*Shape*(*const* Shape& other);

*const* Point& *getOrigin*() *const*;

*const* char\* *getName*() *const*;

*virtual* double *area*();

*virtual* double *perimeter*();

*virtual* void *display*();

    void *display*() *const*;

    double *distance* (Shape& other) *const*;

*static* double *distance* (Shape& the\_shape, Shape& other);

    void *move*(double dx, double dy);

  private:

    Point origin;

    char\* shapeName;

};

#*endif*

//*square.cpp*

#*include* <stdio.h>

#*include* <cmath>

#*include* <iostream>

#*include* "square.h"

using namespace std;

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

double Square::*area*(){

*return* (side\_a \* side\_a);

}

double Square::*perimeter*(){

*return* (4 \* side\_a);

}

double Square::*getSideA*(){

*return* side\_a;

}

void Square::*set\_side\_a*(double side){

    side\_a = side;

}

void Square::*display*(){

    cout << "Square Name: " << *getName*() << endl;

    Point o = *getOrigin*();

    o.*display*();

    cout << "Side a: " << *getSideA*() << endl;

    cout << "Area: " << *area*() << endl;

    cout << "Perimeter: " << *perimeter*() << endl;

}

//*square.h*

#*ifndef* *SQUARE\_H*

#*define* *SQUARE\_H*

#*include* "shape.h"

class Square : virtual public Shape{

    public:

*Square*(double x, double y, double side, *const* char\* name);

        double *area*();

        double *perimeter*();

        double *getSideA*();

        void *set\_side\_a*(double side);

        void *display*();

    private:

        double side\_a;

};

#*endif*

//*circle.cpp*

#*define* *\_USE\_MATH\_DEFINES*

#*include* <stdio.h>

#*include* <cmath>

#*include* <iostream>

#*include* "circle.h"

using namespace std;

Circle::*Circle*(double x, double y, double rad, *const* char\* shapeName) : *Shape*(x, y, shapeName) { radius = rad;};

double Circle::*area*() {

*return* (M\_PI \* *pow*(radius, 2));

}

double Circle::*perimeter*() {

*return* (2 \* M\_PI \* radius);

}

double Circle::*getRadius*()  { *return* radius; }

void Circle::*setRadius*(double r){ radius = r; }

void Circle::*display*()  {

    cout<<"Circle Name: "<<*getName*()<<endl;

    Point o = *getOrigin*();

    o.*display*();

    cout<<"Radius: "<<*getRadius*()<<endl;

    cout<<"Area: "<<*area*()<<endl;

    cout<<"Perimeter: "<<*perimeter*()<<endl;

}

//*circle.h*

#*ifndef* *CIRCLE\_H*

#*define* *CIRCLE\_H*

#*include* "shape.h"

class Circle : virtual public Shape{

  public:

*Circle*(double x, double y, double radius, *const* char\* shapeName);

    double *area*() ;

    double *perimeter*() ;

    double *getRadius*() ;

    void *setRadius*(double r);

    void *display*() ;

  private:

    double radius;

};

#*endif*

//*rectangle.cpp*

#*include* <stdio.h>

#*include* <cmath>

#*include* <iostream>

#*include* "rectangle.h"

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){

         side\_b = sideb;

    };

double Rectangle::*area*(){

*return* (side\_b \* *getSideA*());

}

double Rectangle::*perimeter*(){

*return* (2 \* side\_b + 2 \* *getSideA*());

}

double Rectangle::*getSideB*(){

*return* side\_b;

}

void Rectangle::*set\_side\_b*(double side){

    side\_b = side;

}

void Rectangle::*display*(){

    cout << "Rectangle Name: " << *getName*() << endl;

    Point o = *getOrigin*();

    o.*display*();

    cout << "Side a: " << *getSideA*() << endl;

    cout << "Side b: " << *getSideB*() << endl;

    cout << "Area: " << *area*() << endl;

    cout << "Perimeter: " << *perimeter*() << endl;

}

//*rectangle.h*

#*ifndef* *RECTANGLE\_H*

#*define* *RECTANGLE\_H*

#*include* "square.h"

class Rectangle : public Square{

    public:

*Rectangle*(double x, double y, double sidea, double sideb, *const* char\* name);

        double *area*();

        double *perimeter*();

        double *getSideB*();

        void *set\_side\_b*(double side);

        void *display*();

    private:

        double side\_b;

};

#*endif*

//*curveCut.cpp*

#*include* <stdlib.h>

#*include* <stdio.h>

#*include* <iostream>

#*include* <cmath>

#*include* <string.h>

#*include* "curveCut.h"

using namespace std;

CurveCut::*CurveCut*(double x, double y, double sideA, double sideB, double rad, *const* char\* name):

*Shape* (x, y, name),

*Rectangle*(x, y, sideA, sideB, name),

*Circle*(x, y, rad, name){

*if* (rad > sideA || rad > sideB){

            cerr << "error: radius should be less than the side lengths" << endl;

        }

    }

double CurveCut::*area*(){

*return* (Rectangle::*area*() - Circle::*area*()/4);

}

double CurveCut::*perimeter*(){

*return* (Rectangle::*perimeter*() - 2 \* *getRadius*() + Circle::*perimeter*() / 4);

}

void CurveCut::*display*(){

    cout << "CurveCut Name: " << *getName*() << endl;

    Point o = *getOrigin*();

    o.*display*();

    cout << "Width: " << *getSideA*() << endl;

    cout << "Length: " << *getSideB*() << endl;

    cout << "Radius of the cut: " << *getRadius*() << endl;

}

//*curveCut.h*

#*ifndef* *CURVECUT\_H*

#*define* *CURVECUT\_H*

#*include* "rectangle.h"

#*include* "circle.h"

#*include* "shape.h"

class CurveCut: public Rectangle, public Circle{

    public:

*CurveCut*(double x, double y, double sideA, double sideB, double rad, *const* char\* name);

        double *area*();

        double *perimeter*();

        void *display*();

};

#*endif*

//*graphicsWorlds.cpp*

#*include* <stdlib.h>

#*include* <stdio.h>

#*include* <iostream>

#*include* <string.h>

#*include* "graphicsWorld.h"

#*include* "point.h"

#*include* "square.h"

#*include* "shape.h"

#*include* "rectangle.h"

#*include* "circle.h"

#*include* "curveCut.h"

using namespace std;

void GraphicsWorld::*run*(){

#*if* 1 //*exA*

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

    Point *m* (6, 8);

    Point *n* (6,8);

    n.*set\_x*(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*();

    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;

    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* //*end of block to test array of pointer and polymorphism*

#*endif*

#*if* 1 //*exB*

  #*if* 1

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

    Circle *c* (3, 5, 9, "CIRCLE C");

    c.*display*();

    cout << "the area of " << c.*getName*() <<" is: "<< c.*area*() << endl;

    cout << "the perimeter of " << c.*getName*() << " is: "<< c.*perimeter*() << endl;

    d = a.*distance*(c);

    cout << "\nThe distance between rectangle a and circle c is: " <<d<<endl;

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

    rc.*display*();

    cout << "the area of " << rc.*getName*() <<" is: "<< rc.*area*()<<endl;

    cout << "the perimeter of " << rc.*getName*() << " is: "<< rc.*perimeter*()<<endl;

    d = rc.*distance*(c);

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

    //*Using array of Shape pointers:*

    Shape\* sh[4];

    sh[0] = &s;

    sh[1] = &a;

    sh [2] = &c;

    sh [3] = &rc;

    sh [0]->*display*();

    cout << "\nthe area of "<< sh[0]->*getName*() << " is: "<< sh[0] ->*area*();

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

    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*()<<endl;

    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*()<<endl;

    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*()<<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;

    cc2.*display*();

  #*endif*

#*endif*

}

//*graphicsWorld.h*

#*ifndef* *GRAPHICSWORLD\_H*

#*define* *GRAPHICSWORLD\_H*

class GraphicsWorld{

  public:

*static* void *run*(void);

};

#*endif*