2143 OOP - Test 1

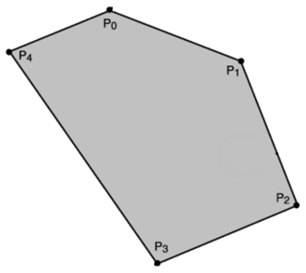
Name: ________________________________

Instructions

- Use pencil only
- Write your name at the top of all pages turned in.
- Staple pages together at the top left corner.
- Make sure your pages are in order, with questions also in order.
- Handwriting that is illegible (messy, small, not straight) will lose points.
- Indentation matters. Keep code aligned correctly.
- Failure to comply will result in loss of letter grade.
- All answers will be written on the paper provided, and not directly on the test.

Background:

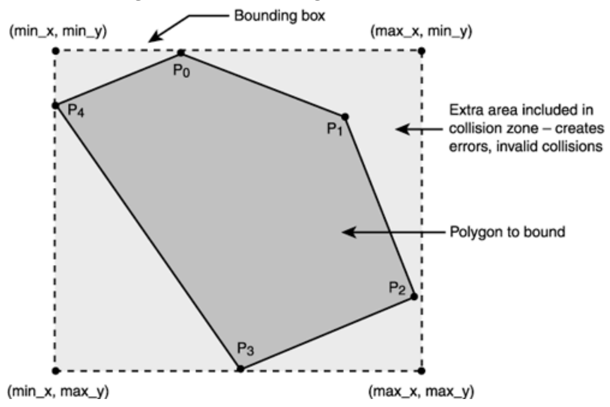
- A polygon is a 2D shape made up of 3 or more sides.



- Each side can also be thought of as a line with a beginning point (x_1,y_1) and an ending point (x_2,y_2).
    - For example, the polygon above has Points: P_0, P_1,P_2,P_3,P_4
    - It also has Lines: (P_0, P_1),(P_1, P_2),(P_2,P_3),(P_3,P_4),(P_4,P_0).
- The length of a line (or distance between two points can be calculated using the following formula:

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- Remember that:
    - Square root is `sqrt(some number)`
    - Exponentiation is `pow(2,3)` or 2^3.
- A bounding box of a polygon can be found by finding the 4 extreme `x,y` values



General

- You are going to write 3 class definitions
- Each class should have methods to set / get the data members of the class.
- Each class definition should build on the previous class.
- Do not implement any methods until asked, definitions only.

Question 1:

What are the 3 Central Principles of OOP?

Answer:

> - Abstraction
> - Encapsulation
> - Inheritance

Question 2 - Point Class

- Write a class that represents a point (x,y).
- X and Y are integer values.

Answer:

```
class Point{
private:
    int x;
    int y;
public:
    Point();             // required
    Point(int,int);      // optional to set x,y when a point is created

    // General Instructions said to include Setters and Getters
    void setX(int);      // required setter
    void setY(int);      // required setter
    void setXY(int,int);// optional setter

    int getX();          // required getter
    int getY();          // required getter

};
```

Question 3 - Line Class

- Write a class that represents a line (x_1,y_1) ,(x_2,y_2).
- Your class should have 2 constructors:
    - One that takes 4 values x_1,y_1,x_2,y_2
    - One that takes 2 values P_1,P_2.

- This class can return the length of its line.

Answer:

```
class Line{
private:
    Point Start;            // required
    Point End;              // required

public:
    Line(int,int,int,int);  // required
    Line(Point,Point);      // required
    double length();        // required

    // General Instructions said to include Setters and Getters

    void setStart(Point);   // one of these setters required
    void setStart(int,int);

    void setEnd(Point);     // one of these setters required
    void setEnd(int,int);

    Point getStart();       // getter
    Point getEnd();         // getter

};
```

Question 4 - Polygon Class

- Write a class that represents a polygon.
- Your polygon can have between 3 and N sides.
- Your class should have multiple constructors:
    - One that initializes an empty polygon
    - One that accepts an array of points [P_1,P_2,...,.P_n].
    - One that accepts an array of lines.
- This class can return the perimeter of the polygon.
- This class can return the area of a bounding box of the polygon.

Answer:

```
class Polygon{
private:
    Line *poly;                 // required data member
                                // to hold sides (lines)

    int numSides;               // size of array of lines

public:
    Polygon();                  // required constructor
    Polygon(Points*,int);           // required constructor
    Polygon(Lines*,int);            // required constructor
    double perimeter();             // required method
    double bboxArea();              // required method

    // General Instructions said to include Setters and Getters

    void setLines(Lines*,int);      // setter
    Line* getLines();               // getter
};
```

Question 5 - Implementation

- Implement the perimeter method of the polygon.

Answer:

```
double Polygon::perimeter(){
    double sum = 0.0;
    for(int i=0 ; i <numSides ; i++){
        sum += poly[i].length();
    }
    return sum;
}
```

Question 6 - Bonus

- Implement the bounding box method of the polygon.

Answer:

```
double Polygon::bboxArea(){
    // Init Min Max vars to be compared to
    int minX = INT_MAX, minY = INT_MAX;
    int maxX = INT_MIN, maxY = INT_MIN;
    // Vars to hold each points values
    int x1,y1,x2,y2;
    // Vars to hold points pulled from a line in the polygon
    Point S,E;


    for (int i=0 ; i < numSides ; i++){
        S = poly[i].getStart();
        E = poly[i].getEnd();
        x1 = S.getX();
        y1 = S.getY();
        x2 = E.getX();
        y2 = E.getY();
        if(x1>maxX) maxX = x1;
        if(y1>maxY) maxY = y1;
        if(x1<minX) minX = x1;
        if(y1<minY) minY = y1;
        if(x2>maxX) maxX = x2;
        if(y2>maxY) maxY = y2;
        if(x2<minX) minX = x2;
        if(y2<minY) minY = y2;
    }

    // Lines to be used to calculate the area (width x height)
    Line Width(Point(minX,minY),Point(maxX,minY));
    Line Height(Point(minX,minY),Point(minX,maxY));

    return Width.length() * Height.length();
}
```