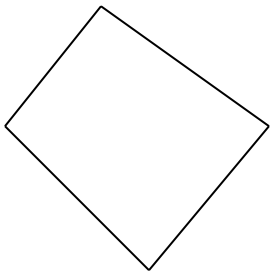
Question 1

Class Concept: Quadrangle (Quadrilateral). (45%)

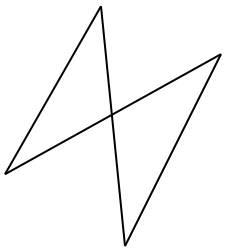
1. (25%) class Quadrangle (quadrilateral). A quadrangle or a quadrilateral is a polygon of 4 sides (with 4 angles). Squares, rectangles, rhombus, trapezoid, parallelogram etc. are the special cases of quadrangles.

A quadrangle Q can be defined via 4 Points in a certain order (just like a Triangle can be defined via 3 Points).

For simplicity, in this question we assume that all the quadrangles you have are of type 1 below which has a real interior, not of the type 2 below, for which two sides cross in the middle. We also assume that all the quadrangles are not of type 3, or the degenerate type, where 3 points are on the same line making this a degenerate quadrangle which is a triangle or a line segment.



Type 1: a normal quadrangle with interior and sides only meet at the endpoints.



Type 2: an abnormal quadrangle for which two sides meet in the middle making the definition of interior and area meaningless.

a) **Define** a class Quadrangle with 4 Points A, B, C, D on the sides as the minimum necessary data.

b) **Add** methods to detect if the quadrangle is a trapezoid, a rectangle, a square, or a parallelogram. ? Note to do this, you need method to compute the angles of the quadrangle.

c) **Add** method to compute the area of a quadrangle.
d) **Test** your program by the following data:

Test Case 1: A (0, 1), B (2, 0), C (0, -1), D (-2, 0)
Test Case 2: A (0, 1), B (1, 0), C (0, -1), D (-1, 0)
Test Case 3: A (0, 1), B (0, 3), C (5, 0), D (-1, 0)

Verify test case 1 is a rhombus and test case 2 is a square *programmatically*. Compute the areas of these quadrangles.

e) Provide your test data to test the other quadrangles such as rectangle, trapezoid, parallelogram etc.

Question 2

2. (20%) Add GUI and graphics to Q1.
(a) (10%) Add GUI: In other words, provide GUI so that you can input the 4 corners of a quadrangle, calculate the area, show the lengths of 4 sides, the angles of 4 interiors.
(b) (10%) Add graphics that draw the quadrangle, especially the special ones like rhombus, trapezoid, rectangle, parallelogram etc.

Question 3

Class LineSegment (60%)

A straight line on the plane is defined by connecting two distinct points or a point and a slope (as you learned in analytic geometry).

When you connect two points, you could create a line segment connecting these two points only or a straight line connecting these two points. For example, if you connect P = (0,0) and Q = (1, 1), you can either think of a line segment of length $\sqrt{2}$, which is the segment PQ, or you can conceive the line y = x of infinite length.

Given a straight line L, a point P not on the line is either above the line or below the line unless that line is vertical or parallel to the y-axis, and in that case, the point P is either on the left or on the right of the line.

Two lines can either intersect or be parallel as you know from 2D geometry. However two line segments can be either parallel, intersect in the middle, intersect at the end, or do not intersect at all. An example of the last case is the line segment PQ for P = (0,0), Q = (1, 1) and line segment RS with R = (-1, 0) and S = (0, -1). PQ is on the line y = x and RS is on the line x + y = 1. The lines y = x and y + x = 1 are perpendicular and intersect, but segments PQ and RS do not intersect.

There is a folder called Supplements, in which there are 4 subfolders Line1, Line2, Line3, and Line4 with code Line.h and Line.cpp for class Line in C++ language.

3. **(25%) Define** a C# class LineSegment, which is defined by two distinct points Point1 = (x1, y1) and Point2 = (x2, y2) (you need class Point), and possibly the slope of the line = (y2-y1)/(x2-x1) (unless x2 = x1 and we have slope = ∞)

(a) **Define** method Length ( ) (the length of segment),

(b) **Define** method Angle () (the angle counterclockwise of how the line segment forms with the x-axis)

(c) **Define** method AboveLine (Point R) to see if a point R is above the line formed from the line segment PQ, and a method BelowLine (point R) to see if a point R is below the line formed from the line segment PQ.

(d) **Define** method onLeft (Point R) if the line is vertical and point R is on the left side. Define method onRight (Point R) similarly

(e) **Define** method parallel (LineSegment L2), meetInTheMiddle (LineSegment L2), meetAtTheEnd (LineSegment L2), doNoTMeet (LineSegment L2) to check if another line segement L2 meets this line segment or not.

(f) **Devise t**est cases that instantiate several line segments LS1, LS2, LS3, etc. and also points P1, P2, etc., so that all methods are used, and these segments may meet or not.

Question 4

4. (10%) Add GUI and graphics to class LineSegment

Question 5

5. (25%) Type 2 Quadrangle
   (a) Devise a type 2 **quadrangle (see Q1)** (you need to show 4 corners A, B, C, and D)
   (b) Use class LineSegment if necessary, prove that your quadrangle is type 2, i.e. line segment AB and Cd meet in the middle
   (c) Add GUI and graphics to clarify this.