CS1115 Fall 2013 Project 7 due Friday, December 6 at 11pm

You must work either on your own or with one partner. You may discuss background issues and general solution strategies with others, but the project you submit must be the work of just you (and your partner). If you work with a partner, you and your partner must first register as a group in CMS and then submit your work as a group. Each problem is worth 5 points. One point may be deducted for poor style. No partners are allowed for the Challenge problem.

1 Background

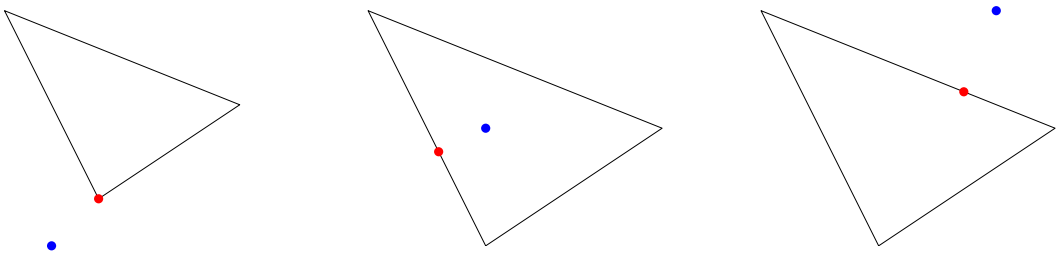
Download the file `Triangle` from the course website. You will find that it is a class definition file. Here is what it looks like showing just constructor and the specifications of its several instance methods:

```
classdef Triangle
% Operations with Triangles
   properties
   % A triangle has three vertices that are Points
        v = Point.empty()
   end

   methods
      % Construct a Triangle object..
      function T = Triangle(P1,P2,P3)
      % P1, P2, and P3 are Points that are to be the vertices of the
      % triangle.
         T.v(1) = P1;
         T.v(2) = P2;
         T.v(3) = P3;
      end

      function Q = Nearest(ThisTriangle,P)
      % P is a point.
      % Q is that point on the perimeter of ThisTriangle that
      % is closest to P.

      function alfa = Include(ThisTriangle,P)
      % P is a point.
      % alfa is true if and only if P is inside ThisTriangle

      function C = InCircle(ThisTriangle)
      % C is the largest circle that fits inside ThisTriangle.

      function C = OutCircle(ThisTriangle)
      % C is the smallest circle that contains ThisTriangle.

      function Q = InCenter(ThisTriangle)
      % Q is the center of the incircle for ThisTriangle.

      function Q = OutCenter(ThisTriangle)
      % Q is the center of the outcircle for ThisTriangle.

      function Show(ThisTriangle,c)
      % c is a character that encodes a color.
      % Displays ThisTriangle in the current figure window assuming that
      % hold is on.

   end % methods
end % classdef
```

The instance methods `InCenter`, `OutCenter`, and `Show` are implemented. You are to implement `Nearest`, `Include`, `InCircle`, and `OutCircle`. The class files `Point.m` and `Circle.m` can be downloaded from the course website and are to be fully exploited. When you are done, there is one file to submit to CMS and that is your fully implemented version of `Triangle`.

2 Nearest

The instance method `Nearest` returns the nearest point on the perimeter of a triangle to a given point. Here are two examples highlighting the given point in blue and the nearest point in red:



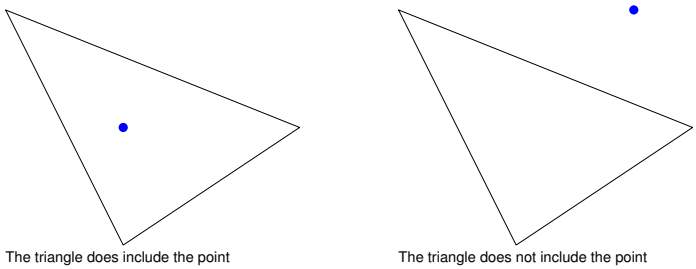
Implement the following instance method:

```
function Q = Nearest(ThisTriangle,P)
      % P is a point.
      % Q is that point on the perimeter of ThisTriangle that
      % is closest to P.
```

Make full use of the methods `Nearest` and `Dist` that are part of the class `Point`.

3 Include

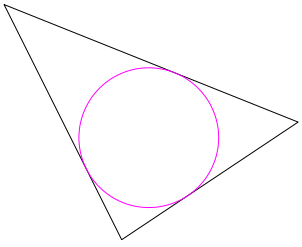
The instance method `Include` is a Boolean function that indicates whether or not a given point is inside a given triangle. Here are two examples:



Make use of the method `DiffSide` that is part of the class `Point`. A test script `ShowNearInclude` is provided on the course website that can be used to check out your implementations of `Nearest` and `Include`.

4 InCircle

Given a triangle, its *incircle* is the largest circle that can fit inside its perimeter. Here is an example



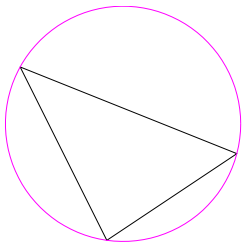
Notice that the circle is tangent to each of the three sides. Implement the following function:

```
function C = InCircle(ThisTriangle)
% C is the largest circle that fits inside ThisTriangle.
```

The instance method `InCenter` can be used to compute the incircle's center. Make effective use of the classes `Point` and `Circle`.

5 `OutCircle`

Given a triangle, its *outcircle* is the smallest circle that encloses it. Here is an example



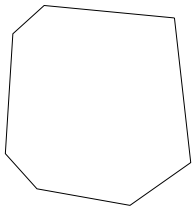
Notice that the circle passes through each of the three vertices. Implement the following function

```
function C = OutCircle(ThisTriangle)
% C is the smallest circle that contains ThisTriangle.
```

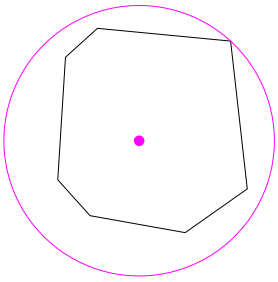
The instance method `OutCenter` can be used to compute the outcircle's center. Make effective use of the classes `Point` and `Circle`. A test script `InAndOut` is provided on the course website that can be used to check out your implementations of `InCircle` and `OutCircle`.

Challenge 7. Intersecting Convex Polygons

A polygon is convex if you never leave the polygon when you walk in a straight line between two points that are in the polygon. Here is an example



Let (x_c, y_c) be the centroid of a convex polygon P. The *bounding disk* for P is the smallest disk that includes P and has center (x_c, y_c). Here is an example:



This problem is about implementing the following class that can be used to answer questions about whether or not a pair of convex polygons intersect:

```
classdef ConvexPoly < handle
% Operations on convex polygons

   properties
      % A convex polygon has n vertices that are points.
      % The bounding circle is a circle that encloses the polygon.
      v = Point.empty();      % A vector of polygon vertices.
      n                       % The number of vertices.
      C = Circle.empty();     % The bounding disk.
   end % properties

   methods
      function P = ConvexPoly(n)
      % Construct a random convex polygon with n sides
         P.n = n;
         tau = 1.5*pi/n;
         theta = linspace(tau,2*pi-tau,n) + tau*rand(1,n);
         x = cos(theta);
         y = sin(theta);
         for k=1:n
            P.v(k) = Point(x(k),y(k));
         end
         % The bounding disk...
         xc = mean(x);
         yc = mean(x);
         r = max(sqrt((x-xc).^2+(y-yc).^2));
         P.C = Circle(r,Point(xc,yc));
      end

      function Translate(ThisConvexPoly,delX,delY)
      % delX and delY are real numbers.
      % For every vertex in ThisConvexPoly, delX is added to the
      % x-coordinate and delY is added to the y-coordinate.

      function Show(ThisConvexPoly,c)
      % Plots ThisConvexPoly
      % The color is specified by the character c.
      % Assumes hold is on.

      function alfa = Disjoint(ThisConvexPoly,ThatConvexPoly)
      % alfa is true if and only if ThisConvexPoly and ThatConvexPoly do
      % not intersect.
```

The constructor and the methods `Translate` and `Show` are implemented. Your job is to complete the class by implementing `Disjoint`. You are free to write "helper methods" to simplify the design of `Disjoint`. A test script `C7` is available on the course website. Submit your fully implemented version of `ConvexPoly` to CMS.

Hints. The key idea is to make effective use of the function `DiffSide` that is part of the `Point` class. Suppose `A` and `B` are convex polygons. Draw a line L through two adjacent vertices in `A`. If no vertex of `B` is on the same side of L as any of A's other vertices, then A and B are disjoint. To make your implementation of `Disjoint` efficient, you should use the fact that A and B are disjoint if their bounding disks are disjoint.