CS W3137 Assignment 6. DUE: Friday, May 2, 11:59 PM

Note: all your work will be handed in electronically. Include a file with your programming submission called "nonprog.txt" that contains the answers (in text) to the non-programming problems below.

Non-programming problems:

1. Problem 7.38 (5 points)
2. Problem 7.43 (5 points)

Programming Problem

1. (90 pts): This exercise will consider THE TRAVELING SALESPERSON PROBLEM - TSP. Here is what you will do:
   (a) (20 points) Write a JAVA program to generate N (x,y) random number pairs with x and y values in the range 0-500. Each of these pairs will be coordinates on an XY grid representing cities locations. Enter the value of N through a GUI text input box and display the points in a window.
   (b) (40 points) Compute the convex hull of the points using Graham's algorithm (see class notes for details on this algorithm). Display the convex hull on top of the points in the window (use a different color for convex hull edges). In Graham's algorithm, you will need to sort the points. You **MUST use Quicksort to sort the points**, and you may use the code in the book for this, but not any Java API sorting methods.
   (c) (20 points) Using the convex hull computed above as a simple cycle of k vertices, compute a TSP tour using the *cheapest insertion metric*: we keep adding vertices that minimize the change in the path's new cost. For example, if we have a an edge (u,v) in the path, we add the vertex x such that:

$$dist(u,x) + dist(x,v) - dist(u,v) \text{ is a minimum} \tag{1}$$

   The tour now replaces edge (u,v) with edges (u,x) and (x,v). We may iterate over all choices of u,v to select this minimum each time. Dislay the final tour by drawing the edges of the tour in a new color, leaving the convex hull edges, and display the total cost of the tour. The web applet for cheapest insertion metric at:

   `http://www-e.uni-magdeburg.de/mertens/TSP/TSP.html`

   gives you an idea of what you need to do (you do not need to replicate this applet exactly with all its options, just use as a guide to what is required).
   (d) (10 points) Add a button to Compute the optimal tour and length and display the length and the tour edges in the window in yet another color. Note: this computation is only realistic for small numbers of $N < 10$.

(5 points) Extra Credit: **2-Opting** add a button that will form a random tour of N points, display the random tour, and then display an animation that highlights pairs of edges that cross and displays their new connections after 2-opting. The web applet also has a solution for this and you can use this as a guide for what is required.