SPRING 2002: **COP 3530** Data Structures
[Programming Assignment 2; Due February 14 in class.]
Traveling Salesperson Heuristics

Problem Description

Your task is to write a program to compute an approximate solution to the traveling salesperson problem. Given a set of N cities with their x and y coordinates, the goal of a traveling salesperson is to visit all the cities (and return home) while keeping the total distance traveled as small as possible. Exhaustive search can help find an optimal tour for small N. For large N, no one knows an efficient method that can find the shortest possible tour for any given set of points, but many methods have been studied that seem to work well in practice, even though they are not guaranteed to produce the best possible tour. Such methods are called heuristics. Note that the length of a tour segment connecting two cities a and b is given by the following distance formula:

$$\text{distance}(a, b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

where the coordinates of a are (x_a, y_a) and that of b are (x_b, y_b). Also note that the length of a traveling salesperson tour is equal to the sum of the lengths of its N segments.

Your program should implement the following heuristic for building a tour incrementally: Start with a three-city tour (from the first city to the second, then to the third, and then back), where the first city is considered as "Home". Now iterate the following process until there are no more cities to process: *Add the next city to the tour by inserting it between two successive cities in the tour at the position where it results in the least possible increase in the tour length.* Note that inserting a new city c between cities a and b results in an increase in the tour length given by the following formula:

$$\text{increase } = \text{distance}(a, c) + \text{distance}(b, c) + \text{distance}(a, b).$$

To implement this heuristic, represent the tour as a circular linked list of nodes, one for each city. Each node in the linked list will contain the coordinates of the city and information (pointer) about the next city on the tour. The linked list should be implemented using the `LinkedList` class provided in `java.util`.

The first line of the input file will contain an integer indicating the number of cities N to be processed. The rest of the input file will contain a sequence of coordinates of cities. Each line of the file will contain the data for one city. Each city will be provided as a sequence of 2 floating point numbers (to be stored as `double` precision real numbers) representing the x and y coordinates of the city. As you read in the cities, you programs should number them from 0 through $N - 1$. These numbers identify the cities now. Data files will be made available to you soon on the course homepage. You should create your own (small) data files to test your program. Your program should output the tour produced by applying the heuristic described above. The tour output by the program should be a list of city numbers. In order to make grading easier, your program should output city number 0 as the first city on the tour.

Submit the source code for your program, the output of `Javadoc`, the tour output by your program, and the length of the tour.

Details

Declare a class called `City` with private data fields called `x`, `y` and `cityNumber`. Implement constructor(s) to initialize the data fields. Also implement methods `getX(), getY(), getCityNumber()` to access the data fields. Declare a method `double distance(City c)` to return the distance from `this` city to city `c`. Finally implement a method `toString()` to print out information about a city.

Declare a class called `Tour` that contains a `LinkedList`. Declare a private data field `double TourLength` to maintain the length of the tour. Also implement a method `double tourLength()` to access the length of the tour. Implement method `int findBestInsertPos(City c)` for finding the best insertion point for a new city `c`. It returns an integer i, which should indicate that the best place to insert the new city `c` is between the i-th city and the $(i+1)$-th city on the current tour. Implement a method called `int insert(City c)` that first calls `findBestInsertPos` and then calls the `add` method from class `LinkedList` to insert city `c` into the tour. It should return the index i where the new city was inserted. Implement a method `updateTourLength(int i)` that updates the length of the tour assuming that a new city has just been inserted after the `i`-th city on the tour. Also implement a method `display()` to print out the tour.

Extensions for the bored

Your program should have appropriate comments describing whatever modifications, additions, and/or improvements you make.

- (Easy) Generalize your program so that it works for points in 3-dimensional space.
- (Moderate) For the 2-dimensional case, get the program to display the tour graphically on the screen as a set of points and line segments.
- (Hard) Use new heuristics to shorten the tour. Any improvements to the program to compute a shorter tour will merit extra credit. One possibility is to avoid tours that cross each other.