COSC 2P93 Prolog Assignment #4

Due date: 12:00 noon Friday March 15
Lates: 12:00 noon Monday March 18, -25%; not accepted afterwards.
Hand in: Printouts of your programs and adequate examples of their execution. Electronic submission of all source code and files. Drawings of best solutions.

Travelling Salesman Problem (TSP)

The TSP is a major research problem in computer science. It is an NP-complete problem, as no polynomial algorithm is known for it. Given a map with cities, a *tour* is a path that starts at some city A, visits all cities exactly once, and then finishes at the same city A. There is a total path distance associated with a tour. The TSP problem is to find the tour with the smallest overall distance.

You are to write a Prolog program that searches for as small a tour as possible. Here is some advice on how to proceed:

a. You should represent the cities using an appropriate data structure in Prolog, eg.

```
city(Name, Long, Lat).          % Long and Lat are the X and Y coords on map
```

b. You might represent a tour as a permutation of city names. For example, if there are 5 cities, then here is a tour: [1, 5, 2, 3, 4]. The start city is 1. You can measure the total distance of the tour by summing the Euclidean distance between cities 1 and 5, then 5 and 2, 2 and 3, 3 and 4, and finally, 4 and 1. (Look up the (X,Y) coordinates of each city using the data structure in (a)).

c. You will have to use some sort of search algorithm. Some options are:

(i) An exhaustive search will look at all permutations, and find the one(s) with the smallest tour distance. It guarantees an optimal solution. However, it might take too long to run!

(ii) A random (blind) search looks at random permutations until some time limit (eg. total # tours tested) is reached, and reports the best tour found. It might not be optimal, but at least you will get a solution in a reasonable amount of time.

(iii) A simple heuristic search is the "hill climber". It is a little more sensible than the blind search. It works as follows:

```
BestTour = random permutation of cities
Repeat until (time limit expired): {
    NewTour = (Swap 2 cities in BestTour)
    If dist(NewTour) < dist(BestTour)
    Then BestTour = NewTour
}
Report the BestTour
```

d. Feel free to use the Sicstus "random" library. Look at the Sicstus manual for how to use facilities in it.

e. Hand-draw your best solution(s). Compare it to "known" solutions (see next point). Hand-in your drawings.

f. Test your program on 2 data sets (both available on 2P93 web site):

(1) A 4-by-4 square grid of cities. (see 2P93 web site)
(2) The Djibouti data set (from the following link...):

http://www.tsp.gatech.edu/world/countries.html