COS128: General Computer Science — Honors	**Spring 1996**
Problem Set # 4	
Handed out:Feb. 29, 1996	*Due: March. 13, 1996*

In this assignment you will write a program that tries to solve the famous *Travelling Salesman Problem.* In view of the exam next week and midterms week the following week, we have kept this assignment simple. You also have two weeks to do the assignment. However, when budgeting your time, keep in mind that you will receive another assignment next Thursday (March 7) due March 27 (the Wednesday after spring break).

In the *Travelling Salesman Problem,* ("TSP") the user gives the program n points in the plane. The program's goal is to find the shortest tour that starts from the first point, visits every point exactly once, and then returns to the first point. This problem comes up in many applications. Let us just mention the application from which it derives its name. Suppose a Princeton salesman has a list of n clients, who live in various towns in New Jersey. The salesman would like to drive away from Princeton, visit every client once, and return to Princeton at the end. Clearly, he would like to save time and fuel, so he would like to make this tour as short as possible. Finding an optimum tour for the salesman thus amounts to solving an instance of TSP.

It is known (in a certain sense that will be made more precise later in the course), that *no* program can find the optimum tour in some reasonable length of time — say, a few days — for modestly large n on even the fastest computers. Therefore for many decades, researchers have been trying to write programs that do not necessarily compute an optimum tour, but only a near-optimum tour (say, one whose length is at most 10% more than the length of the optimum tour).

In this assignment we give you the English description of such a program — called the *double nearest neighbor heuristic* — which you must then convert into C code.

Warmup — the nearest neighbor heuristic:

Before considering the *double nearest neighbor heuristic*, let us consider a simpler method of which it is an extension: the *nearest neighbor heuristic.* The nearest neighbor heuristic finds a tour as follows. It starts at the first point, and extends the tour to the point closest to this point. From there it extends the tour to the point closest to the second point. The program continues this way, at each step extending the current partial tour to include the point that is not yet in the tour and which is nearest to the point added at the previous step.

Now for some practical details. Note that a point in the plane is represented by an ordered pair (a, b), where a is its x-coordinate and b is its y-coordinate. Furthermore, the distance between two points (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Warmup assignment — optional:

Write a C program that finds a salesman tour according to the nearest neighbor heuristic. The output of the program should consist of: (i) The length of the tour (ii) The sequence of points encountered along the tour. The output in part (ii) should be properly formatted, putting (say) 10 points per line of output. Print the index of each point rather than it's coordinates, i.e. i for the i'th point.

We are providing you with a partially-written program, shown below, that already shows you how to represent the data. (This program fragment can be found in file `/u/cs126/progs/tspprog1.c`.)

```
#include <stdio.h>
#include <stdlib.h>       /*contains srandom() and random()*/
#include <math.h>         /*contains sqrt()*/
#define N 100
double x[N];
double y[N];
void initialize()
{
  int i;

  srandom(1);    /*set seed for random numbers */
  for(i=0; i<N; i++)
    {
      x[i] = 97*((double) (random()%231)/231);
      y[i] = 97*((double) (random()%231)/231);
    }
}
/* define your functions here */
void main()
{
/* define your variables here */

initialize();
/* put in your code after this point */
}
```

Each point is represented by its x and y coordinates, each of which is a floating point number. The symbolic constant `N` represents the number of points. The set of points will be stored in an two arrays, `x[]` and `y[]` of type `double`. The intended meaning is that the x-coordinate of the i'th point is `x[i]` and the y-coordinate is `y[i]`.

The partially-written program also contains a function `initialize()` that initializes the `x` and `y` arrays using a random number generator.

Note: You can store the partial tour in an array

```
int tour[N]
```

where `tour[j]` contains the index of the point that is added to the tour at the j'th step.

The double nearest neighbor heuristic:

Like the *nearest neighbor heuristic*, the *double nearest neighbor heuristic* also maintains a partial tour and at each step extends the partial tour to a point that has not been visited yet. The difference is that whereas the nearest neighbor heuristic can extend the current partial tour only at the endpoint that was just added, the double nearest neighbor heuristic can extend the current partial tour at either of the two endpoints. That is, the heuristic searches among all the points that are not yet in the tour, and adds the one that is closest to either of the end-points.

The assignment:

Write a C program that finds a salesman tour according to the double nearest neighbor heuristic. Use the data representation and partial-written program as discussed above for the nearest neighbor heuristic. Produce the same output as requested in the warm-up assignment: (i) The length of the tour and (ii) the sequence of points encountered along the tour. The output in part (ii) should be properly formatted, putting (say) 10 points per line of output. Print the index of each point rather than it's coordinates, i.e. i for the i'th point.

Hint: There is an elegant way to implement the double nearest neighbor heuristic using only arrays and no advanced data structures.

Testing your program:

By varying `N` and the argument given to the function `srandom` (initially 1, change to some other positve integer), you can produce different numbers of points with different coordinates. You can also design specific tests by replacing the body of `initialize` with an assignment of coordinates to points as you choose. Vary the number and location of points to test your program. However, submit the version with `N = 100` and function `initialize` as given above.

Output:

To view your program's output more easily, especially when the number of points is large, you may wish to direct it to a file. To direct your program's output into a file `result`, you type `a.out > result`. Then you can examine the file `result` using emacs.

Submission:

You need only submit your implementation of the *double nearest neighbor heuristic*. Put your program in a file called `doublenearest.c`. To electronically submit it, type

```
/u/cs128/bin/submit 4 doublenearest.c
```

If you have trouble implementing the double nearest neighbor heuristic, you may submit an implementation of the nearest neighbor heuristic (the "warm-up") for partial credit. The deadline for electronic submission is **10pm on Wednesday, March 13**, and for hardcopy submission is in lecture on **Thursday, March 14**.

Extra credit:

The most straightforward way to get the *exact* solution to an instance of the TSP is to do *exhaustive search*, that is, to try all $n!$ different tours and compute the total length for each. Write a program to do this exhaustive search. You will not be able to run your program for very large n, but see how large you can make n without running for more than 15 minutes. Compare an exact solution found by exhaustive search with the tour found by the double nearest neighbor heuristic for the same set of points.

Hint: Generating the $n!$ different tours can be organized recursively.