Computer Science I Program #2: Tree House Walking (Recursion)

Please check Webcourses for the Due Date

Read all the pages before starting to write your code

There are several trees (an even number) with treehouses in your backyard and you have pitched the bright idea to your parents that it would be fun to connect pairs of tree tops with a rope ladder, so that friends in one tree house could get to friends in another tree house without going back to the ground.

Naturally, it would only be fair if each of the tree tops was connected to exactly one another.

Your parents aren't too psyched about the daunting physical task of erecting the rope ladder connections. They've decided that they will only do the project if you can match pairs of tree tops in such a way that the total distance of rope ladders is minimized.

Assume that the length of a rope ladder built between treetops located at (x_i, y_i) and (x_j, y_j) is

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Write a program to determine this minimum sum of rope ladder distances so that your treehouse utopia is realized! (Note: A valid configuration of rope ladders contains *n* ladders to connect 2*n* trees and each tree is connected by exactly one rope ladder.)

The Problem

Given the (x, y) positions of 2*n* treehouse, of all possible ways to create *n* pairs of distinct trees, find the minimum possible sum of distances between each tree in the pairs.

The Input (to be read from standard input)

The first line will contain a single positive integer, *c* ($c \le 25$), representing the number of test cases to process. The test cases follow.

The first line of input for each case will contain a single positive integer, *n* ($n \le 8$), representing that your backyard has 2*n* trees total.

The following 2*n* lines of each input case will contain a pair of space separated positive integers, x_i and y_i, representing that the ith treetop is located at the coordinate (x_i, y_i), with $-10000 \le x_i, y_i \le 10000$.

The Output (to be printed to standard out)

For each input case, output a single floating point number rounded to exactly 3 decimal places, representing the minimum sum of distances possible if the rope ladders are built between pairs of trees.

Sample Input
```
2
1
19 -18
16 -14
3
0 10
10 0
10 10
15 15
0 0
-5 -5
```

Sample Output
```
5.000
28.284
```

Implementation Restrictions/ Run-Time/Memory Restrictions
1. For full credit, your algorithm must run in O(n(2n)!/2^n) time. **In particular, any correct solution which runs in O(n(2n)!) time, which tries upto 16! permutations of 16 distinct objects will NOT receive full credit.**

2. There are three tiers of credit for this program. If you can get the code to run fast enough for n = 5, then that's worth 70% of the points. If you can get it to run fast enough for n = 6, that's worth 90% of the points, and for full credit, it has to run fast for n = 8.

The intended run-times are O((2n)!) for n = 5, O ((2n)!/2^n) for n = 6, and O ((2n)!/ (2^n * n!)) for n = 8.

Note: It's completely okay if you don't find a solution that runs fast enough for n = 8. My expectation is that students will be able to solve the problem for n = 5. I am curious to see how many can solve it for n = 6 and n = 8. Both of these require modifications to the permutation code which skip over redundant permutations.

3. For full credit, the function you write to try each possible way of pairing up treetops **must be recursive.**

4. You must only declare your important variables **INSIDE** the case loop.

Deliverables

You must submit one file over WebCourses:

1) A source file, *treehouse.c*. Please use stdin, stdout. There will be an automatic 10% deduction if you read input from a file or write output to a file.

2) A file describing your testing strategy, *lastname_Testing.doc(x) or lastname_Testing.pdf*. This document discusses your strategy to create test cases to ensure that your program is working correctly. If you used code to create your test cases, just describe at a high level, what your code does, no need to include it.

3) Files *treehouse.in* and *treehouse.out*, storing both the test cases you created AND the corresponding answers, respectively. **(Note: 5 of your own test cases are expected and all of them can be hand made, where you "eyeball" the answer.)**