COP 3502C Programming Assignment # 2

Recursion

Read all the pages before starting to write your code

Overview

This assignment is intended to make you work with recursion and some permutation. The code should not be lengthy. However, solving this problem could be challenging. - don't wait until the weekend it's due to start it!

Your solution should follow a set of requirements to get credit.

Please include the following commented lines in the beginning of your code to declare your authorship of the code:

/* COP 3502C Assignment 1

This program is written by: Your Full Name */

Compliance with Rules: UCF Golden rules apply towards this assignment and submission. Assignment rules mentioned in syllabus, are also applied in this submission. The TA and Instructor can call any students for explaining any part of the code in order to better assess your authorship and for further clarification if needed.

Caution!!!

Sharing this assignment description (fully or partly) as well as your code (fully or partly) to anyone/anywhere is a violation of the policy. I may report to office of student conduct and an investigation can easily trace the student who shared/posted it. Also, getting a part of code from anywhere will be considered as cheating.

Deadline:

See the deadline in Mimir and webcourses. No late submission will be accepted. An assignment submitted by email will not be graded and such emails will not be replied according to the course policy.

What to do if you need clarification on the problem?

I will create a discussion thread in webcourses and I highly encourage you to ask your question in the discussion board. Maybe many students might have same question like you. Also, other students can reply and you might get your answer faster. Also, you can write an email to the TAs and put the course teacher in the cc for clarification on the requirements.

How to get help if you are stuck?

According to the course policy, all the helps should be taken during office hours. Occasionally, we might reply in email.

Problem Description: **Tree House Walking**

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 $2n$ trees and each tree is connected by exactly one rope ladder.)

The Problem

Given the (x, y) positions of $2n$ 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 in.txt file)

The first line will contain a single positive integer, c ($c \leq 25$), 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 ($n \leq 8$), representing that your backyard has $2n$ trees total.

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

The Output (to be printed as well as to be written in out.txt file)

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	**Sample Output**
2 1 19 -18 16 -14 3 0 10 10 0 10 10 15 15 0 0 -5 -5	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 four files over mimir:

1) A source file, *main.c*.

2) A file describing your testing strategy, ***lastname_Testing.txt*** ~~*(Changed from .docx/.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) You have to submit one test file test.txt that has at least two test cases. You also need to submit testout.txt file that is generated by your code while testing based on your test.txt file. **(Note: Hand made test cases should be fine. It's not necessary to make a max case to get full credit here.)**

Rubric (subject to change):

According to the Syllabus, the code will be compiled and tested in Mimir Platform for grading. If your code does not compile in Mimir, we conclude that your code is not compiling and it will be graded accordingly. We will apply a set of test cases to check whether your code can produce the expected output or not. Failing each test case will reduce some grade based on the rubric given bellow. If you hardcode the output, you will get -200% for the assignment. Note that we will apply more test cases while grading. So, passing the sample test cases might not guarantee that your code will also pass other test cases. So, thoroughly test your code.

1. If a code does not compile the code may get 0. However, some partial credit maybe awarded. A code having compiler error cannot get more than 50% even most of the codes are correct
2. Not using recursion will receive 0

3. There is no grade for a well indented and well commented code. But a bad indented code will receive 20% penalty. Not putting comment in some important block of code -10%
4. There will be some grade for your test cases and testing strategy
5. There will be grade for proper permutation implementation
6. There will be significant amount of grade if your code works for n=5 (~75%)
7. To get a better grade, your code needs to run fast for n = 6 (~90%)
8. To get 100%, your code needs to work fast for n=8 as well.

Some hints:

- **Read the problem completely and try to draw the example points provided in the sample input and see how the result is calculated from those points, which points are finally connected to minimize the distance.**
- **Think, how can you use permutation and the "used" array approach to keep track which tree is used in this process**
- **Get more concept from the lab problem "Dance Recital" and its solution. This uses the "used" array for a different problem scenario and it will help you a lot to solve the assignment problem.**
- **Start your coding by loading the test cases and data and make sure you code is able to read them properly before processing.**
- **Read a test case and process it and then read the next one, etc.**
- **Do not wait till the end to test your code.**
- **Do not hesitate to take help during all of our office hours.**

Good Luck!