COP 3502C Programming Assignment # 3

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 assignment also intended to explore the use of ChatGPT for some optimization. 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 3

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 on 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: Connecting Garages

Students at Monster University have found the parking information system helpful and enjoyed the recent free parking spot game. However, they often experience frustration when they enter a garage only to discover that many parking spots are blocked due to an upcoming event or for mysterious reasons. As a result, they are forced to search for available spaces in other garages. The journey between garages is time-consuming, primarily because of the numerous traffic signals on public roads, causing students to be late for classes or exams.

In response to several complaints, the parking garage manager has proposed building dedicated expressways between garages to allow vehicles to move from one garage to another without needing to navigate public streets. Ideally, each garage would be connected to exactly one other garage. However, constructing these expressways is costly. Therefore, Monster Campus has decided to move forward with the project only if the following conditions are met:

1. There are exactly **n expressways** connecting **2n garages**.
2. Each garage is connected to **only one expressway**.
3. The garages must be paired in a way that minimizes the **total distance** of the expressways.

Assume that the distance of an expressway built between garages 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 expressway distances so that the campus managements are satisfied!

The Problem

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

The Input (to be read from standard input using scanf (do not use file i/o. Using file i/o will get zero)

The first line of input will contain a single positive integer, *n* ($n \leq 8$), representing that there are *2n* garages in total.

The following 2*n* lines will contain a pair of space separated positive integers, x_i and y_i, representing that the i^{th} garage is located at the coordinate (x_i, y_i), with $-10000 \leq x_i, y_i \leq 10000$. Additionally, the line will also contain the name of the garage (a single word string with maximum 20 characters).

The Output (to be printed in standard console output (do not use file i/o. Using file i/o will get zero))

Output a single floating point number rounded to exactly 3 decimal places, representing the minimum sum of distances possible if the expressways are built between pairs of garages. After printing the sum, print n lines with the pairs garage names and their distance that gave the minimum sum of distance. The output has to be in the following format: (fromGarageName, toGarageName, distance in exactly 3 decimal places). The "from garage" should be the garage that appeared first in the input compared to the "to garage". The order of the garage names should match the order they appeared in the input. If multiple sets of pairs result in the same minimum distance, print only the set that comes first in the permutation based on the input order. Ensure your output exactly matches CodeGrade's output.

Sample Input 1	**Sample Output 2**
1 19 -18 GARAGEA 16 -14 GARAGEB	5.000 (GARAGEA, GARAGEB, 5.000)
Sample Input 2	**Sample Output 2**
3 0 10 GARAGEA 10 0 GARAGEB 10 10 GARAGEC 15 15 GARAGED 0 0 GARAGEE -5 -5 GARAGEF	28.284 (GARAGEA, GARAGEB, 14.142) (GARAGEC, GARAGED, 7.071) (GARAGEE, GARAGEF, 7.071)

Implementation Restrictions

1. You must use the permutation "used" array technique in your solution
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 80% of the points. If you can get it to run fast enough for n = 6, that's worth 95% of the points, and for full credit, it has to run fast for n = 8.
3. For most part of the code like for n=5, you are not allowed to use ChatGPT. You must use the permutation code discussed in the class and modify it to solve this problem. However, for optimizing the code for n=6 or more, you must explore how *ChatGPT* can help you to optimize your code and you must submit a report with your finding that should contain:
    a. Your code that runs fast enough for n=5
    b. Screenshot of the prompts and response from ChatGPT that helped you to improve your code to run faster with n=6 or more.
    c. A short paragraph with few lines explaining how ChatGPT was able to helped you to optimize your code.
4. It's completely okay if you don't find a solution that runs fast enough for n = 8. You will not lose much credit. 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.
5. Your permutation must be a recursive code and should be based on the used array technique even after optimization.
6. You are allowed to use only two global variables. No other global variables are allowed:
    a. One for array of garages
    b. One for the final set of permutation int array that stores the index of the garages that gave you the minimum distance

Deliverables

You must submit four files over codegrade:

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

2) A file describing how you have used ChatGPT to optimize your code for *n>=6*, ***yourlastname_qpt.pdf***. The content of this document discussed in the restrictions 3 above.

Rubric (subject to change):

According to the Syllabus, the code will be compiled and tested on codegrade Platform for grading. If your code does not compile on codegrade, 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. So, submit your code even if it does not work.
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 grade for proper permutation implementation
5. There will be significant amount of grade if your code works for n=5 (~80%)
6. To get a better grade, your code needs to run fast for n = 6 (~95%)
7. To get 100%, your code needs to work fast for n=8 as well.
8. If you don't submit the required pdf file, 8 pts will be deducted if you get more than 95

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 garage is used in this process (We have discussed how to perform permutations for any set objects in the permutation lecture!. Think about, permutation of a string "CAT" could be achieved by creating permutation of the numbers from 0 to 2 (as the length of the string is 3). If you get a permutation 1, 2, 0, it means the permutation is "ATC", as 1 is the index of A, 2 is the index of T and C 0 is the index of C)
- Start your coding by loading the test cases and data and make sure you code is able to read them properly before processing.
- Do not wait till the end to test your code.
- A good idea would be writing a function for calculating distance between two garages based on their index in the parameters
- Do not hesitate to take help during all of our office hours.

Good Luck!