Individual Design Assignment
(CS Students)
CSE-3310 002/003
Rev C 04/03/2020

This assignment will evaluate a student against the ABET criteria SO-C

- "an ability to design a system, component, or process to meet desired needs ."

References:

https://en.wikipedia.org/wiki/Geographical_distance
ttps://openflights.org/data.html

Problem:

Given a list of all airports in the world (provided from the file "airports.dat"), determine all airports withing a given radius of a given airport.

Write a command line program that accepts as input the ICAO identifier (example "KDFW" is Dallas-Ft Worth) of an airport, and a radius in miles, and returns a list of the closest airports, in order, one airport per line (ICAO code, airport name[in double quotes], distance in miles). The program will continue accepting input until a negative radius is entered (in this case, an ICAO code needs to be entered but it is ignored). Prior to exiting, the program will print the time spent in initialization in seconds, the average time in calculating a result in seconds, and the bytes taken by the programs data structure(s).

Example (note the results are not 'real', just representative of the desired output):

% **./airports**

KDFW 20.0

KDAL "Dallas Love Field" 5.32

KNFW "NAS Fort Worth JRB/Carswell Field" 12.75

XXXX -99.9

0.00011 0.00010 250000

%

For this assignment, you will design and implement a software subsystem that provides this functionality.

It is expected that the program will execute in three distinct phases:

- initialization - data structures are created
- operation - where input is processed
- shutdown - any cleanup tasks are performed

You are allowed to use any functions in the C library or the C++ Standard Template Library. All other routines you must write.

Reliability is desired in this program. It is suggested that all error conditions that can occur be checked for and either corrected or identified to the user without exiting the program.

In case you were wondering, "no, the world is not flat". The specific algorithm used is up to the student, but the specific results need to be in the requirements.

The following limitations exist:

- Dynamic memory allocation only allowed during initialization (prior to the first input position processing)

Objectives (in order of importance):

- Correct answer.
- Minimize the time required to return an answer (average processing time). (note: the amount of time between program start and the processing of input { the 'initialization time'} is not of concern).

The source code, with makefile, needs to be turned in as a tar file. The code will be compiled and ran prior to grading. No credit for this assignment will be given unless the source code compiles, without warning or errors. The program must also be reliable enough to evaluate it against the requirements.

The written portion of this assignment is a single MS word document, containing the following sections:

1. Problem Statement
   A concise statement of the problem and the objectives. (couple of paragraphs)
2. Schedule
   Provide a list of all of the tasks to complete this assignment, an estimate of the hours required, **and the actual hours expended.** A table would be ideal for this. You will not be graded on the amount of time expended, but on the realism and detail of the planning process.
3. Requirements Analysis
   Using the assignment and provided information, specify the requirements in clear expressive statements using 'shall' and 'will'. Ensure each requirement is noted to be either 'Functional' or 'Non Functional'. (expectation of between 5 and 10 requirements). Some of

these requirements will be the algorithm you come up with to solve the problem. **Ensure that the strategy that you are using to meet the requirements and constraints is clearly stated; if needed add a few paragraphs making it clear.** It is acceptable to record any assumptions that are not stated in this assignment as requirements.
Provide a table showing all functional requirements, and the class/ method in your design that will implement them.

4. Design
   Describe the design of your solution, using a textual description along with diagrams. At a minimum, this will include a class diagram. In the textual description, focus on the run time behavior and how the top level objectives will be met.
5. Alternatives
   Briefly describe two alternate designs, how they would work and why they were not selected. These alternatives can describe designs that would not meet the requirements or constraints.
6. Constraints
   Describe all of the constraints and explain how the design accommodates them. Note there are many constraints due to the problem itself, not just the ones listed on the assignment.
7. Source code
   Will be graded from the tarball.
8. Analysis
   Design and execute a set of test case(s) to verify the program meets requirements. For the 'average processing time' requirement, present the results of execution in a graph. Describe your analysis of the graph and how it shows that the requirement is met. Provide the inputs, expected results, observed results, and the reason for selecting each input in a table.

Turn in two files:

- an msword document, named "firstName_lastName.doc" (where your name is substituted).
- a tarball of the source code (not compressed), named ""firstName_lastName.tar".

Students that do not follow these conventions will receive no credit for this assignment.

NOTE: The rubric to be used for the majority of the assignment is shown in the table below.

NOTE ! NOTE: Due to the criteria in the rubric, many students will score lower on this assignment than they expect. A '3', or an average score results in a 60% vs. the usual 70%. Very few students will score in the 'A' range for this assignment.

Suggestion:

- Use strtok() to parse the input from the file.
- Use sizeof() to determine the size of your main data structures.

This is an individual assignment. It is not a group assignment. Do not work on this project in a group. Do not share code.

Grading rubrics for SO-c/C

	Excellent (5 pts)	Good (4 pts)	Satisfactory (3 pts)	Poor (2 pts)	Unacceptable (1 pt)
Problem Statement	Problem statement is clear and shows full understanding of the realistic constraints.				Problem statement is NOT clear and it does not show understanding of the realistic constraints.
Requirements Analysis	Clearly explains the overall design objectives; clearly defines functional and non-functional requirements; analyzes the constraints and limitations of the design project.				Very few design objectives and functional and non-functional requirements defined. Constraints and limitations inadequately or not defined.
Alternatives	Design alternatives are considered and evaluated				Design alternatives are not considered or they are not clear.
Constraints	Real world limitations and constraints such as economic, ethical, health, and safety are considered.				Constraints are not properly considered and/or they are not realistic
Tools and Skills	Design demonstrates evidence of the ability to use appropriate tools, and skills to deliver the final system				There are no evidence of the ability to use appropriate tools, and skills to deliver the final system.
Documentation	Documentation is complete, well organized, and clear in purpose. includes goals, timeline, schematics.				Documentation is not complete and it is not organized.
Management	Design is broken down into manageable tasks; all time estimates are reasonable; all resources are assigned to all tasks.				Design is not broken into manageable tasks and time estimates are not reasonable.