Hack 13.0

Computer Science I
Searching & Sorting

Department of Computer Science & Engineering
University of Nebraska–Lincoln

Introduction

Hack session activities are small weekly programming assignments intended to get you started on full programming assignments. You may complete the hack on your own, but you are *highly encouraged* to work with another student and form a hack pair. Groups larger than 2 are not allowed. However, you may discuss the problems *at a high level* with other students or groups. You may not share code directly.

If you choose to form a Hack Pair, you *must*:

1. Both join a hack pair on Canvas (go to People then Hack Pairs)
2. You must both work on the hack equally; it must be an equal effort by both partners. Do not undermine your partner's learning opportunity and do not undermine your own by allowing one parter to do all the work.
3. Turn in only one copy of the code under the individual whose last name comes first (with respect to Canvas).

You are graded based on style, documentation, design and correctness. For detail, see the general course rubric.

Category	Point Value
Style	2
Documentation	2
Design	5
Correctness	16
Total	**25**

Table 1: Rubric

Correctness:

- 4 points for student tester; they need 5 airports
- 12 for official test suite

Problem Statement

You will use the implementation of your Airport data model from a previous hack to develop several reports that will require you to sort and search (a subset of) the International Civil Aviation Organization database for particular airports.

We have provided an updated header file, `airport.h` with the specific functions that you need to implement. First, you'll need to implement 8 different comparator functions for your `Airport` structure. The details of the expected orders are documented in the header file. The file is available directly here.

Second, you'll need to write code that produces several reports. All your code needs to be placed into the following function:

```
void generateReports(Airport *airports, int n);
```

which takes an array of `Airport` structures and produces the following reports which should all be output to the standard output.

- To help you troubleshoot, you should print out all the structures in the original order to the standard output.
- Sort the airports by each of the 8 comparators and print them out (8 reports total).
- Search for and print out the airport in the array that is closest (via air distance) to Lincoln. Lincoln is located at 40.8507N, 96.7581W.
- Search for and print out the airport that is the geographic west-east median[1] of the given airports with respect to longitude.
- Search for an airport located in the city New York (city would be `New York` and the country would be `US`) and print it out if it exists. If no such airport exists, print out an appropriate message.
- Search for an airport whose type is `large_airport` and print it out if it exists. If no such airport exists, print out an appropriate message.

[1] With 0-indexed arrays, the median is usually the element at index `n/2` when sorted. This relies on truncation to give the middle index of odd-sized arrays and prefers the "right" element for even-sized arrays.

Instructions

- Place all of your function definitions in a source file named `airport.c` and hand it in with your header file, `airport.h`. You may add any utility functions you wish but you must *not* change any of the signatures of the required functions.
- In addition, you must create a main test driver program that demonstrates your reports with an array that contains at least 5 airports. Name this file `airportReport.c` and hand it in.
- You are encouraged to collaborate any number of students before, during, and after your scheduled hack session.
- You may (in fact are encouraged) to define any additional "helper" functions that may help you.
- Include the name(s) of everyone who worked together on this activity in your source file's header.
- Turn in all of your files via webhandin, making sure that it runs and executes correctly in the webgrader. Each individual student will need to hand in their own copy and will receive their own individual grade.