COMP 1602, Computer Programming II

Assignment #4

Date Due: April 12, 2013

Description

This assignment requires you to develop a system for *Fly High Airways* that will enable customers to find out about the different cities it flies to, and the distance in miles between the cities it flies to, etc.

(a) A file, `cities.txt`, contains information on various cities of the world. Each line of data contains data for one city as follows, in the order given:

- the name of the city
- the three-letter airport code for that city (used by airlines around the world)
- the country in which the city is located
- the population of the city

The amount of lines of data in the file is unknown beforehand but data is terminated with a line containing a city name of "END" only. Sample lines of data are as follows:

Name of City	**Airport Code**	**Country**	**Population**
Port-of-Spain	POS	Trinidad-and-Tobago	400000
Miami	MIA	USA	2000000
Toronto	YYZ	Canada	3000000
New-York	JFK	USA	5000000
…	…	…	…
END			

(b) *Fly High Airways* flies to cities all around the world. The file, `flights.txt`, stores data on the cities the airline flies to. Each line of data contains the three-letter airport code of the city from which the flight leaves (its origin), the three-letter airport code of the city to which the flight goes to (its destination), the distance between the two airports, the type of aircraft used for the flight, the capacity of the flight, and a number indicating how often the flight is made per week (0: daily, 1: bi-weekly, 2: weekly).

If the airline does not have a flight between two airports but one of its partners does, the distance between the two airports is expressed as a negative number. Data is terminated with the word "END" on a line by itself. Sample lines of data are as follows:

Origin	Destination	Distance	Aircraft	Capacity	Frequency
POS	MIA	1621	Boeing-767	250	0
POS	YYZ	-2517	Boeing-767	240	1
MIA	YYZ	1486	Airbus	175	0
…	…	…			
END					

From the data above, it can be seen that the airline does not fly between POS (Port-of-Spain) and YYZ (Toronto) but one of its partners does. Also, the flight from POS to MIA is made on a daily basis.

(c) Declare a struct *City* to store the information on each city and write a function, *readCities*, which accepts a one-dimensional array of *City* structs as a parameter and reads the data from `cities.txt` and stores it in the array. The function should return the amount of cities that was read from the file.

(d) Declare a struct *Flight* to store the information on each flight (except for origin and destination) and write a function, *readFlights* which accepts a two-dimensional array of *Flight* structs as well the one-dimensional array of *City* structs, and the amount of cities in the array, as parameters.

If there is a flight between two cities, *origin* and *destination*, the location of each city in the one-dimensional array of *City* structs is found. Suppose *origin* is found in location *i* and *destination* is found in location *j*. At the [*i*][*j*] location of the two-dimensional array, the flight information (distance, aircraft, capacity, and frequency) should be stored.

(e) Write a *main* function which declares *cities* as a one-dimensional array of *City* structs. Assume that there are no more than 100 cities. The cities are then read using the *readCities* function from part (c).

The *main* function must also declare *flights*, a two-dimensional array of *Flight* structs. The array must be initialized and then *readFlights* must read the data from `flights.txt` into the *flights* array.

(f) After reading the data from the `cities.txt` and `flights.txt` files, your program must generate the following menu:

1. Query for City by Name
2. Query for City by Airport Code
3. Query for a Direct Flight between Two Cities
4. Query for 1-Stop Flights between Two Cities
5. Query for Flights from a City
6. Query for Flights to a City
7. Display Statistics
8. Quit

The following table describes the functionality that must be provided for each menu option:

Menu Option	**Functionality**
1	The user must input the name of a city and the relevant information for that city must be displayed (by searching the *cities* array).
2	The user must input the three letter airport code of a city and the relevant information for that city must be displayed (by searching the *cities* array).
3	The user inputs the three letter airport code of an *origin* city and a *destination* city and ***if there is a direct flight*** between these two cities, the relevant information must be displayed. The *cities* array must be searched to find the corresponding locations in the *flights* array.
4	As in Option 3, the user inputs the three letter airport code of an *origin* city and a *destination* city. If there is a city, *middle*, such that there are flights from *origin* to *middle* and then from *middle* to the *destination* city, this is a ***one-stop*** flight. All such one-stop flights must be displayed.
5	The user inputs the three letter airport code of an *origin* city and all the direct flights leaving that *origin* city should be found and displayed.
6	The user inputs the three letter airport code of a *destination* city and all the direct flights arriving at that *destination* city should be found and displayed.
7	When this option is selected, the length of the longest direct flight of the airline (not its partners) is displayed. Also, the total miles flown by the airline per week must be displayed. Note that if an airline makes a daily flight from POS to MIA (distance = 1621, frequency = daily), the distance travelled per week is 1621 * 7 (between these two cities alone).
8	Quit the program

Programming Guidelines

(1) The *cities* array is a one-dimensional array of *City* structs (which are read from the `cities.txt` file).

(2) Suppose there is a flight between POS and MIA (which is obtained from the `flights.txt` file). The locations where POS and MIA are stored in the *cities* array must first be found. Suppose these locations are *i* and *j*, respectively. To indicate that there is a flight between POS and MIA in the flights two-dimensional array, flights[*i*][*j*] must contain the relevant information pertaining to the flight (distance, aircraft, capacity, and frequency). There is no need to store the origin and destination cities.

(3) Three-letter airport codes input by the user must be checked to ensure they are valid before attempting to find flights.

(4) Note that flights[*i*][*j*] where *i* = *j* represent flights between the same cities. Assume that there are no such flights.