CSCI 2170 OLA 5
Electronic submission due midnight, Sunday November 23rd

The FlyWithUs airline company would like for you to help them develop a program that generates flight itinerary for customer requests to fly from one city to another city. To support flight itinerary generation, it is necessary to build a database of all available flights.

For this assignment, you are to write a program that builds an adjacency list structure to store flight information. An adjacency list is an array of linked lists. Your program needs to have two classes, a sorted-list class (implemented in ola4) and a flightMap class. The flightMap class implements the adjacency list structure.

Your client program reads in a list of city names for which the company currently serves. The city names are in the data file named "cities.dat". Then, it reads in a list of flights currently served by the company. The flight information is in the data file "flights.dat". Here is the format of these two data files:

cities.dat : First line of the data file gives the total number of cities served by the company. Next, the names of cities the airline serves, one name per line, for example:

31 ← total number of cities served by the company
Albuquerque
Chicago
San-Diego
…

flights.dat : each flight record contains the flight number, a pair of city names (each pair represents the origin and destination city of the flight) plus a price indicating the airfare between these two cities, for example:

178	*Albuquerque*	*Chicago*	*450*
703	*Chicago*	*Atlanta*	*120*
550	*Nashville*	*San-Diego*	*580*
833	*Chicago*	*Washington-DC*	*500*
1180	*Atlanta*	*Chicago*	*120*
...			

After reading and properly storing the information, you program should print out the flight map/information in a well-formatted table:

Origin	Destination	Flight	Price
From Atlanta to:	Chicago	1180	$120
	New-York	320	$180
	Seattle	1200	$210
From Chicago to:	Atlanta	703	$120
	Washington-DC	833	$500
	…		

Copy the data files into your own account by:

```
ranger$ cp ~cen/data/cities.dat  cities.dat
ranger$ cp ~cen/data/flights.dat  flights.dat
```

Since we are to store flight records as data in the linked list, the listItemType for the linked list with be a flight record structure. Define the struct in a type.h and a type.cpp file as following:

- Create a type.h and type.cpp file to define the list item type:
  - type.h defines the FlightRec structure and the overloaded operators (==, >, and <, <<) for this structure
  - type.cpp implements the overloaded operators

The adjacency list will be implemented in a FlightMap class. Implement the **Flight Map** ADT(flightMap.h and flightMap.cpp) which has the following data and at least the following methods:

- Data

        - number of cities served by the company
        - list of cities served by the company – (use a 1D array for this. you should create/allocate memory for this array dynamically)
        - flight map of the company stored in the form of an adjacency list, e.g., array of sortedListClass objects. (The array needs to be created dynamically)
    - constructor(s) and destructor
    - methods:
        - reads cities (cities.dat)
        - reads flight information and build the adjacency list (flights.dat)
        - displays the flight information as shown above.

- Implement the client program that:
    - Creates a flight map object
    - Reads the list of cities
    - Reads flight info and builds the flight map, i.e., the adjacency list
    - Print the flight map in a formatted table as shown above

Instructions to submit your program

- Create a script file by following the steps below:

  First, navigate to the directory where your program source file is located, then follow the steps below:

```
ranger$ script log5
ranger$ pr –n –t –e4 type.h
ranger$ pr –n –t –e4 type.cpp
ranger$ pr –n –t –e4 sortedListClass.h
ranger$ pr –n –t –e4 sortedListClass.cpp
ranger$ pr –n –t –e4 flightMap.h
ranger$ pr –n –t –e4 flightMap.cpp
ranger$ pr –n –t –e4 ola5.cc
ranger$ aCC type.cpp sortedListClass.cpp flightMap.cpp –o run
ranger$ run
ranger$ exit
```

- **Electronically submit the program with the following command:**

handin ola5 type.h type.cpp sortedListClass.h sortedList.cpp flightMap.h flightMap.cpp ola5.cc log5