CSCI 2170 Spring 2006
OLA 5 (Part A (50 pts) Due Thursday March 23rd,
Part B (50 pts) Due Thursday March 30th, 2006)

The EastWest airline company wants you to help them develop a program that generates flight itinerary for customer requests to fly from some origin city to some destination city. But first, they need to have an efficient way of storing and maintaining their database of all available flights of the company. First, your program should read in a list of city names for which the company currently serves. The list of names can be read from a data file named "cities.dat". Then, your program reads in a list of flights currently served by the company. The flight information can be read from the data file "flights.dat".

cities.dat : the names of cities that EastWest airline serves, one name per line, for example:

16 ← 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*	*250*
703	*Chicago*	*San-Diego*	*325*
550	*Nashville*	*San-Diego*	*180*

…

Copy the data files into your own account by : frank% cp ~cli/data/cities.dat cities.dat
frank% cp ~cli/data/flights.dat flights.dat

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

	Origin	Destination	Flight	Price
	From Atlanta to:	Chicago	1180	$89
		New-York	320	$180
		Seattle	1200	$210
	From Chicago to:	Atlanta	1181	$89
		WashingtonDC	3400	$67
		…		

Program requirements:

- Implement a pointer based **ListClass**. This list class should keep records/nodes in ascending order of the city name. **Thoroughly test this class to make sure that all methods work correctly before moving on to the rest of the project.**
- Implement a FlightMap ADT, which has the following data and at least the following operations:
  - Data
    - Number of cities served by the company
    - list of cities served by the company – you should create this array dynamically
    - flight map of the company stored in the form of an adjacency list, e.g., array of ListClass objects. (The array needs to be created dynamically)
  - constructor(s) and destructor
  - operations
    - read cities (cities.dat)
    - read flight information and build the adjacency list (flights.dat)
    - display the flight information as shown above.

Part A of the project:

- Implementation of the sorted **ListClass**. It should include
  - Constructors and destructor

    - Inserts a flight record in ascending order by **destination city**
    - Deletes a flight record that matches with origin and destination cities as parameter values
    - Finds and returns a flight record when provided with a origin and destination city
    - Print the entire list
    - Returns the length of the list
    - Returns whether the list is empty
- A client program to test the listClass. This client program should :
    - Create a listClass object
    - Read the flight record, one record at a time until the end of file, from **flights.dat**, and insert each record into the **list** in ascending order or the **destination city.**
    - Print the list of records (one record per row)
        - It should include an output that display the number of records in the list
    - Find and print the flight that matches user supplied origin and destination city.
    - Delete the record that matches the user supplied origin and destination city
    - Print the list of records (one record per row)

What to turn in for Part A:

```
Frank%script log5
Frank%pr –n –e4 listClass.h
Frank%pr –n –e4 listClass.cpp
Frank%pr –n –e4 ola5A.cc
Frank%aCC listClass.cpp ola5A.cc –o run
Frank%run
Frank%exit
```

Part B of the project:

- Implement the **FlightMap** ADT as described above
- Implement the client program that:
    - Create a FlightMap object
    - Read cities
    - Read and build flight map
    - Print flight map

What to turn in for Part B:

```
Frank%script log5
Frank%pr –n –e4 listClass.h
Frank%pr –n –e4 listClass.cpp
Frank%pr –n –e4 flightMap.h
Frank%pr –n –e4 flightMap.cpp
Frank%pr –n –e4 ola5.cc
Frank%aCC listClass.cpp flightMap.cpp ola5.cc –o run
Frank%run
Frank%exit
```