CSCI 2170 OLA 5 Spring 2012

Part A (100 pts) Electronic submission and hard copy due beginning of class, Monday, April 2nd,

Part B (100 pts) Electronic submission and hard copy due beginning of class, Monday April 9th,

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 and a flightMap class. The sorted-list class implements the pointer based ADT sorted-list, that is a slight variation as the one discussed in class. The flight 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:

```
16              ← 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
```

<u>Part A of the project:</u>

Implement a <u>pointer based</u> **sortedListClass**. This 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.**

The **sortedListClass** should include at least the following member functions:

- o Default constructor and copy constructor
- o Destructor
- o Inserts a flight record in ascending order by **destination city**
- o Deletes a flight record that matches with origin and destination cities as parameter values
- o Finds and returns a flight record when provided with a origin and destination city
- o Print the entire list
- o Returns the length of the list
- o Returns whether the list is empty

- A client program to test the sortedListClass. It should have at least the following:
  - o Create a sortedListClass object
  - o Read the flight record, one record at a time until the end of file, from **flights.dat**, and insert a record into the list in ascending order of the **destination city.**
  - o Print the list of records (one record per row)
    - ▪ It should include an output that display the number of records in the list
  - o Find and print the flight that matches user supplied origin and destination city.
    - ▪ Prompts the user to enter origin and destination city. If there is flight going from origin to destination, print the entire flight information, otherwise, say "no flight available".
  - o Delete a flight record that matches the user supplied origin and destination city from the list. <u>Perform three delete operations</u>. Each time, prompts the user to enter origin and destination city. If the flight between origin and destination city exists, delete the flight record from the list:
    1. For the first delete operation: enter origin and destination correspond to that of the first flight record in the list;
    2. For the second delete operation: enter origin and destination correspond to a flight record in the middle of the list;
    3. For the third delete operation: enter origin and destination that does not have a corresponding flight record in the list.
  - o Print the list of records (one record per row) → this prints the records in the final list.

For this assignment, you are required to:

- Create a type.h and type.cpp file to define the data type
  - o type.h defines the FlightRec structure and the overloaded operators (==, >, and <, <<) for this structure
  - o type.cpp implements the overloaded operators
- Create a makefile to compile your program. Refer to handout and instruction from the class.

<u>Instructions to submit your program</u>

- o **Hard copy**:
  - ▪ 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 log5A
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 ola5A.cc
ranger$ pr –n –t –e4 makefile
ranger$ make
ranger$ run
ranger$ exit
```

    Enclose the hardcopy of the program and the program evaluation sheet in a folder
- o **Soft copy:**
  - ▪ login the ranger system with www.cs.mtsu.edu/nx,

    - login to PeerSpace through the web browser provided by the ranger system, click on *tools|Assignments* to submit your softcopy.

Part B of the project:

- 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

- **Hard copy**:
    - 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 log5B
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 ola5B.cc
ranger$ pr –n –t –e4 makefile
ranger$ make
ranger$ run
ranger$ exit
```

      Enclose the hardcopy of the program and the program evaluation sheet in a folder
- **Soft copy:**
    - login the ranger system with www.cs.mtsu.edu/nx,
    - login to PeerSpace through the web browser provided by the ranger system, click on *tools|Assignments* to submit your softcopy.