CSCI 2170 Spring 2013
OLA 7 (due midnight, Monday, April 29th)

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. For each customer request, indicate

- Whether a sequence of EastWest flights from the origin city to the destination city exists. Give appropriate message if it does not exit;
- If an itinerary exists, the actual flight itinerary, and the price of the entire flight itinerary.

This program will build on the code written for OLA5. The three data files used in the program are:

(1) **cities.dat** : the names of cities that airline serves, one name per line, for example:

30
Albuquerque
Chicago
San-Diego
.

(2) **flights.dat**: pairs of city names (each pair represents the origin and destination of one of the flights) 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*
...

(3) **requests.dat** : pairs of city names, each pair represents a request to fly from some origin to some destination, for example:

Albuquerque *Chicago*
Chicago *San-Diego*
Nashville *Seattle*
San-Diego *New-York-City*
...

Copy the files again since the data files are updated for this assignment.

ranger% cp ~cen/data/filename filename

The program should produce output of the following form:

Request is to fly from Albuquerque to San-Diego.
EastWest airline serves between these two cities.
The flight itinerary is:

Flight #	*From*	*To*	*Cost*
178	*Albuquerque*	*Chicago*	*$250*
703	*Chicago*	*San-Diego*	*$325*
		Total:	*$575*

Request is to fly from Albuquerque to Paris.
Sorry, EastWest airline does not serve Paris.

Request is to fly from San-Diego to Chicago
Sorry, EastWest airline does not fly from San-Diego to Chicago.

You are required to:

- Implement pointer based StackClass
- Add the following functions and data to the FlightMap class created in OLA 6:
  - The non-recursive **IsPath** algorithm discussed in class used to find the itinerary between two cities. ***Modify the code to display the full itinerary if one is found***.
  - Additional functions used by the IsPath function:
    - MarkVisited: record that a city has been visited; void MarkVisited(string city)
    - IsVisited: returns true if a city has been visited; bool IsVisited(string city)
    - UnvisitAll: mark all the cities as not visited; void UnvisitAll()

- GetNextCity: pass back the first adjacent city that has not been visited. If such a city is found, return true, otherwise, return false; bool GetNextCity(string city)
- GetCityNumber: returns the index of a city when the name of the city is provided; (You may have already implemented this as a LinearSearch)
- GetCityName: returns the name of a city when the index of the city is provided.

- Additional data for the class: ***visited*** array to record whether a city has been visited during the itinerary planning process. This array should also be allocated dynamically. (Hint: do this in ReadCities method)
- Add overloaded assignment (=) operator to sortedListClass
- Create copy constructor for the flightMapClass.

To turn in the program, follow these steps:

- **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 log7
ranger% pr –n –e4 type.h
ranger% pr –n –e4 type.cpp
ranger% pr –n –e4 sortedlistClass.h
ranger% pr –n –e4 sortedlistClass.cpp
ranger% pr –n –e4 stackClass.h
ranger% pr –n –e4 stackClass.cpp
ranger% pr –n –e4 flightMap.h
ranger% pr –n –e4 flightMap.cpp
ranger% pr –n –e4 ola7.cc
ranger% aCC type.cpp sortedListClass.cpp stackClass.cpp flightMap.cpp ola7.cc  -o  run7
ranger% run7
ranger% exit
ranger% lph log7
```

- **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.