ECE 2574 ♦ Introduction to Data Structures and Algorithms
Homework 3

Overview: The stack is one of the most commonly used data structures. For this assignment, you must write a C++ program that implements the ADT stack using non-circular, singly linked lists, and then use it to solve a graph search problem: Flight Map Search.

Objective: The main goal of this assignment is to gain experience in implementing the *ADT stack* and using it to solve problems. You will also gain more experience with various linked lists.

Program specifications: Implement an ADT version of the class `Stack`. Use the ADT to develop a C++ program that (1) reads a flight map that is annotated with flying costs and an origin city from an input file, (2) determines a flight route to each city that can be visited from the origin city and the associated total cost, and (3) for each reachable city, writes the flight route from the origin city to the city and the associated total cost to an output file.

Consider the following flight map as an example:

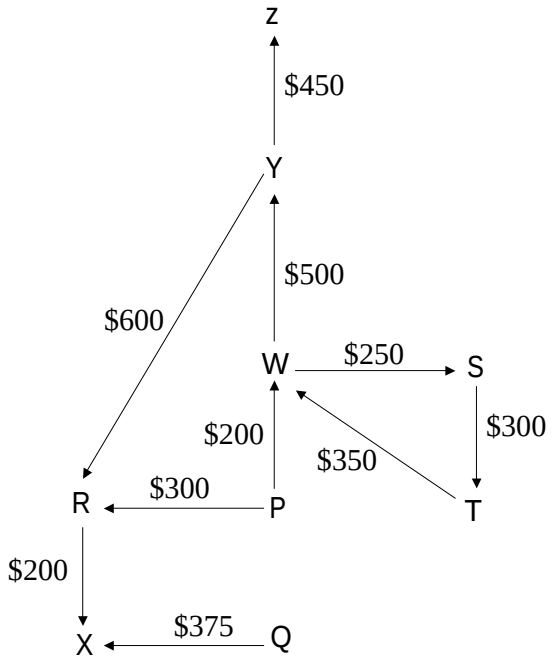

Figure 1: An example flight map.

In this map, letters represent cities, arrows represent direct flights between cities, and dollar amounts written beside the arrows indicate the associated flying costs. For example, there is a direct flight from city P to city W and the cost of this flight is $200.

Given such a flight map, we say that city B can be *visited* from city A, if there exists a flight route (a set of direct flights) from city A to city B. For example, the cities R, X, W, S, T, Y, and

Z can be visited from the city P, whereas the city Q cannot be visited from city P. The set of cities that lie in the flight route from city P to city Z is: P, W, Y, and Z.

If city B can be visited from city A, we define the *total cost* for flying from A to B as the sum of the cost of all direct flights in the flight route from A to B. For example, the total cost for flying from city P to city Z (flight route: P, W, Y, Z) is $200 + $500 + $450 = $1150.

Your program should be invoked as follows:

`searchmap` *`input_file output_file`*

The format of the input file is as follows:

input_file

```
P
P W 200
P R 300
R X 200
Q X 375
W S 250
S T 300
T W 350
W Y 500
Y Z 450
Y R 600
```

The file **input_file** contains a description of an origin city (P) and a cost-annotated flight map (which is shown in Figure 1). The first line of the file contains the name of the origin city. Each subsequent line describes a pair of cities that has a direct flight between them and the associated cost of the direct flight. For example, the second line indicates that there is a direct flight from city P to city W and the cost of this direct flight is $200.

The file **output_file** shown below contains the desired output of the program. Each line of the output file displays a (destination) city that can be visited from the origin city, the flight route from the origin to the city, and the associated total cost.

output_file

Destination	Flight Route from P	Total Cost
R	P, R	$300
X	P, R, X	$500
W	P, W	$200
S	P, W, S	$450
T	P, W, S, T	$750
Y	P, W, Y	$700
Z	P, W, Y, Z	$1150

If multiple routes exist to a destination city, your program is only required to find *one* of those routes, and that route does *not* necessarily need to be the cheapest route.

File naming conventions: The name of your source files should be *searchmap.cpp, mapclass.h, mapclass.cpp, stackP.h,* and *stackP.cpp*.

The file *searchmap.cpp* contains the "client" code that uses a *mapclass* ADT. The *mapclass* ADT stores a flight map and provides operations that facilitate a search of the flight map. The specifications and implementations of the *mapclass* ADT are contained in the files *mapclass.h* and *mapclass.cpp*, respectively.

The *mapclass* ADT internally uses the stack ADT to facilitate a search for cities for which routes are not known. The header file *stackP.h* contains the specification of a stack ADT, and the file *stackP.cpp* contains the implementation of the stack ADT. To gain experience with linked-lists, you are required to write *your own* stack ADT using non-circular, singly-linked lists. You should not directly use the StackP module that is given in the Carrano textbook, although you can use it for inspiration. Also, you should not use the C++ STL stack to develop your program.

Assumptions: You can assume that only a single character will be used to represent a city. You should not make any assumptions on the total number of cities or direct flights between pairs of cities that the flight map will have. You are required to implement the stack ADT using a non-circular, singly-linked list. Also, you may assume that the input and output file names will appear on the command-line in the order described earlier.

Exception handling: Your implementation should catch the following exceptions:

1. If input and/or output filenames do not appear on the command line, or the input file does not exist.
2. Bad input file. This includes an input file that contains any of the following errors:
    (a) No origin city
    (b) A city represented with more than one character
    (c) A pair of cities with no associated flight cost
    (d) A pair of cities with a non-positive flight cost
    (e) A malformed pair of cities (e.g., a city pair with just 1 city, or 3 or more cities)
3. If *new* returns NULL when allocating memory.

What to hand in: After your program is complete, upload your 5 files (*searchmap.cpp, mapclass.h, mapclass.cpp, stackP.h, stackP.cpp*) to Scholar. Be careful to use these file names, to test your files before uploading them, and upload the correct files. The files that you upload to Scholar are the files that will be graded.

Partial credit: If you could not get your program to run, also turn in a written description (MS Word or PDF file) that describes 1) the problems you encountered, 2) what parts of your program work correctly, and 3) suggestions to the grader as to how your program may be tested for partial credit consideration. If your program runs, you do not need to turn in such a file.

Grading: Refer to the Grading Guidelines document.

Honor Code: You must work independently on this assignment. Please review the statement in the syllabus.