Programming Assignment #6

Stacks and Queues and Roller Coasters

CS 2308.003 and 004 Fall 2018
Instructor: Jill Seaman

Due: Monday, 12/3/2018: upload electronic copy by 9:00am.

Stack:

Given the class declaration of a Stack in the provided file **Stack.h**, use a stripped-down version of the StringList class from Programming Assignment #5 (also provided in files **StringList.h** and **StringList.cpp**) to implement the stack operations. This means that in the definitions of the Stack functions you will need to call some of the StringList functions (add(str), remove(int) etc.). Hints: pick the appropriate end of the StringList to be the top of the stack, and keep track of the number of elements in the stack.

- Please inline your Stack function definitions (place the code in the Stack.h file).
- You may add private members to Stack.h
- DO NOT CHANGE StringList.h OR StringList.cpp!!!
- Test your implementation with the provided driver (StackDriver.cpp).

Queue:

Repeat the same exercise for the Queue: Use the class declaration in the provided file **Queue.h**, and use the same unchanged StringList class files to implement the queue operations. Test your implementation with your own driver.

Roller Coaster:

Write a program that uses the Queue to simulate the management of the line of people waiting to ride a roller coaster in an amusement park. Here are the assumptions:

- The roller coaster seats x people.
- The riders come in groups that have a name and a size.
- The group size will not be more than x (the number of seats on the train).
- Groups cannot be split into separate runs of the roller coaster (all must go on the same train).
- Groups/people cannot jump ahead in line.
- A coaster train must be as full as possible before it can leave the station.

Input/Output:

The program input will be available in a file named "riders.txt". The first line of the file contains the number of seats the roller coaster contains. The following lines contain a name and a number, each corresponding to the name of the group and the number of people contained in the group.

The output should have the following format:
For each train, which train it is (#1, #2, etc) with a listing of the riders on that train.
For each rider, output the group name and a number for that member (see the expected output for an example).
Finally, output the total number of riders and the total number of train runs:
Total Riders: 42
Total Trains: 5
See the sample input (riders.txt) and output (expected.txt) in the provided files.

NOTES:

- This program must be done in a **Linux or Unix** environment, using a command line compiler like g++. Do not use codeblocks, eclipse, or Xcode to compile.

- **Hints**: Put the "names" of the riders in the queue. When the next group will not fit in the queue, that train is full, and the queue can be emptied and output. Note that some of the trains will not be full.

- Use of a Queue is required for full credit (though you may be able to solve the problem without using a Queue).

- If you decide to pass the Queue to a function, you should pass it by reference.

- You can use the function to_string(int) (from C++11) to convert an int to a string, and the + operator to append two strings together.

Logistics:

For this assignment you need to submit **three** files: Stack.h, Queue.h and your driver. Submit the files using the Assignments tool in TRACS before the deadline.

For this assignment, no printout is required! The feedback cover sheet will be posted to the assignments tool in TRACS (or emailed to you) after the assignments are graded (and the grade will be posted to the TRACS gradebook tool).

See the assignment turn-in policy on the course website (cs.txstate.edu/~js236/cs2308) for more details regarding penalties.