Assignment DS6

Due Date: November 26 @ 4 PM

Purpose

The purpose of this assignment is to use queues and inheritance (it's back!) in a fun (as always!) manner.

Problem

You have the slightly-stressful job of air traffic controller. As such, you have the responsibility of keeping planes from crashing into one another on runway 42L. This runway is used for both landing and taking off, hopefully not at the same time. You decide the best way to take care of things is to write a nifty program in C++.

Input

This will be a menu-driven program. A menu will always be shown on the screen, giving the following choices:

ct	clear flight for takeoff
cl	clear flight for landing
qt	queue flight for takeoff
ql	queue flight for landing
s!	stop the program

The controller will always type two letters. Any combination of two letters not specified above should result in an error message. In any case, the program should continue and show the output screen (see below) and then the menu again.

In the case of `qt` or `ql`, the user must type in the airline, which is **up to** 3 characters, and the flight number, which may be **up to** 4 digits in length.

Output

After each user input, the screen should display the following information:

```
-------- Runway 42L --------
Status: TWA  873 landing
    or
Status: TWA  873 taking off
    or
Status: clear

Landing queue:
  1.  UA    39

Takeoff queue:
  1.  AA  8912
  2.  NW   292
  3.  UA   340

 ...menu here...
```

The first status is for the case of `ct` or `cl` with the flight taken from the appropriate queue, the second for `qt` or `ql` with the newly added flight inserted into the proper queue. You may wish to add additional status messages to handle cases such as when the controller types in `ct` and there are no planes in the takeoff queue.

Specifics

- Each flight is identified by an up to 3-letter airline and up to a 4-digit flight number (true fact: the larger the flight number, the smaller the plane).
- You must use the STL queue to manipulate the landing and takeoff queues.
- You can have additional queues as needed. However, you should not have any other data structures (arrays, etc.) to store lists of planes.
- The queues can be of any length (have you ever been to O'Hare airport?). However, the display screen is rather small. Therefore, you should only display the first 5 flights in the queue, even though there may be more flights stored in the queue (those flights will be displayed eventually as flights leave the queue). If there are more planes in the queue, there should be a message indicating that fact.
- Your program should include a base class that defines a runway object. Two other classes, one for a takeoff object and the other for a landing object, should be derived from the base class. Each class should include *only* those elements that pertain to that object. Private, public, and protected elements should be used as appropriate.
- You must write a class function (method) called `showQueue()`. Its job is to display, at most, the first five elements in a queue that is passed in as a parameter. The function should not destroy the original queue. This method should be defined in the base runway class and called from each of the derived classes; there's no sense in writing essentially the same function twice!
- The general menu/loop structure should be done in a function that is not part of the hierarchy described above. The program should *use* the various classes to solve the problem as part of this function.
- All input menus and output data should be neatly arranged and **aligned**.

Notes

The program does not have to make sure planes do not crash into each other; it merely does what the user commands.[1] If the user specifies 3 takeoffs in a row, then that is what the program should do.

While planes may crash, your program should not (if, for example, the controller selects `ct` and there are no planes in the takeoff queue).

Comment your program as previously specified. Functions/methods should be short (< 20 lines).

Submission: As usual, archive all of your files in one zip file. Submit this file via Canvas with the usual naming conventions; for example, my submission would be `gousieDS6.zip`.

There is **no** hard copy to turn in for this project.

It's not reality, but that's the way it is.
– Brooks Robinson

[1] To do this, we would have to add in a time component to determine if a runway was still in use.