
- No late tutorials will be accepted.
- If there are specific instructions for making a function, please follow them exactly. That means that
  - function names
  - function return types
  - parameter types and order

  should all be **EXACTLY** as described. If the script can't read it, you will receive 0 for that part.
- Your **Tutorial 6** code will be marked by a Python script. You are being given the script so that you may make sure your code runs correctly. As such, submissions with improper files, configuration, or function signatures will not be accepted.

---

1 Submission Instructions

Download "tutorial6.zip". Unzip it into your working directory. There is a directory "tutorial6" and the test file "t6test.py". In the "tutorial6" folder are "Student.h", "Student.cc", "test.cc" and "defs.h" to get you started. To the "tutorial6" folder you should add the following files.

1. Header and source files for the `Queue` class described below.
2. A Makefile.

You will zip the "tutorial6" directory into a file "tutorial6.zip". If you are doing this in the course VM you must do this from the command line. Open a terminal in the folder that contains "tutorial6". Use the command `zip tutorial6.zip tutorial6`. This will zip the `tutorial6` folder, or update it if you change the contents. Submit `tutorial6.zip` to Brightspace by the deadline. DO NOT USE .tar OR .tar.gz FILES. Use .zip only please.

2 Testing Your Tutorial With `t6test.py`

`t6test.py` is a test script that is very similar to what will be used to mark your tutorial (basically I will change the input and expected output ... unless I get lazy, then I won't change anything). So the mark you see here should be the mark you receive (as long as you did not hard code output. If you try a shortcut you might get burned). To run `t6test.py`, open a command line in the directory containing `t6test.py`. You may have to make it executable, so type `chmod +x t6test.py`. You may run the script as is, in which case it will look for a file to unzip. Or, if you have not zipped your files yet you may supply a "-nozip" argument, in which case it looks for the "tutorial6" folder.

To have the script unzip `tutorial6.zip` and then test your code, run `./t6test.py`. To skip the unzip step use `./t6test.py -nozip`. When your tutorial is being officially marked we expect a zipped file.

Running this script will generate a file "results.txt" just outside of the "tutorial6" folder. This will have some useful output as well as the mark.

3 Learning Outcomes

In this tutorial you will learn how to make the `Queue` class, which is a cousin to the `LinkedList` class.

4 Instructions

4.1 Overview

In this tutorial you will write the `Queue` class. This will be the same `Queue` class from Assignment 3, Section 5.4 EXCEPT it will have a different data member. You will be able to use this `Queue` class in your Assignment 3 by changing all occurrences of `Student*` to `WHLocation*`.

4.2 Queue Class

Complete Section 5.4 in Assignment 3 using `Student` in place of `WHLocation`. The instructions from Assignment 3, Section 5.4 are repeated below with the substitution made for your convenience.

This has a similar structure to the `List` class we saw in class. It would make sense to make `Queue` a `List` subclass. However, we have not learned about templates yet, and in this application `Queue` and `List` use different classes as data. So unfortunately we must keep them separate classes, which means that you will be copying a lot of code from the `List` class to put into `Queue`.

1. Nested class - make a private nested class `Node`. You may use the `Node` class from the `List` class, however, change the `data` to type `Student*`.
2. Member variables:
   (a) `Node* head` - same function as the `head` variable in the `List` class.
   (b) `Node* tail` - similar to `head` except `tail` should always point to the last element in the `Queue`. This will make it easy to add elements to the back of the `Queue`.
3. Constructor - initialize both `head` and `tail` to `NULL`.
4. Destructor - Delete all `Node`s in the `Queue`. DO NOT destroy the data.
5. Member functions:
   (a) `isEmpty()` - return true if the `Queue` is empty.
   (b) `void peekFirst(Student** loc)` - return the `Student*` data from the first location if it exists, or assign `NULL` to `*loc` otherwise. DO NOT delete the `Node`.
   (c) `void popFirst(Student** loc)` - return the `Student*` data from the first location if it exists, or assign `NULL` to `*loc` otherwise. Delete the `Node` if it exists.
   (d) `addLast(Student* loc)` - Add `loc` to the end of the `Queue`.

4.3 Makefile

Your Makefile should compile two object files, `Location.o` and `StoreLocation.o`. It should link these object files to the `test` executable. In addition your Makefile should contain an `all` command that creates the `test` executable and a `clean` command that removes all executables and object files.

4.4 t6test.py

Run this python script to test the classes described above. Correct all errors.