Project Stack and Queue

Data Structures CSCI 2320

Project Objective

Learn how to use a Stack and Queue ADT in C++ to solve a problem.

Project Overview

Your mission is to implement a **BearFlix** movie application that will save a list of movies you want to watch and a list of movies you have watched. You will be given a list of movies you have watched called `movie_history.txt`, which should be managed as a `Stack`. You will also be given a list of movies you want to watch called `movie_queue.txt`, which should be managed as a `Queue`. You will be given a partially implemented `main.cpp` and a full implementation of `Stack.h` and `Queue.h`.

Project Requirement

Verify that you are using the ISO std 20+ compiler. See Blackboard for more information.

Project Tasks

Review Task

- Clone the starter repository to your computer.
- Review the implementation of `Stack.h` and `Queue.h`. Recall that a `Stack` is a Last In First Out data structure (LIFO) and a `Queue` is a First In First Out data structure (FIFO).

Visual Studio 2022 Users

- Create your Visual Studio 2022 Project and add the provided `main.cpp` file as well as the `Stack.h` and `Queue.h`.
- Copy the following files to the same location as your project/solution files.
  - `movie_history.txt`
  - `movie_queue.txt`

VS Code/Mac Users

- You will need to run your program from the Terminal because it accepts input. As always, please let me know if you have questions.

Develop your main driver

In the partially implemented main driver `main.cpp` implement the following using the provided `movieHistory` (Stack) and `movieQueue` (Queue).

- When a user enters `a`, prompt the user for a movie name and Add the movie to the queue.
- When a user enters `w`, move a movie from the Watch queue to the stack history.
- When a user enters `d`, Delete the next movie from the queue.
- When a user enters `h`, print the movie History in stack order.
- When a user enters `r`, print the most Recently watched movie.
- When a user enters `q`, print the entire movie Queue.
- When the user enters `n`, print the Next movie to watch.
- When the user enters `x`, eXit the program. This code is provided.

Handle errors in your main driver

- When appropriate, show the following message to standard output and let the user try again.
  Movie queue is empty.
- When the user enters an invalid command, print the following error to standard output and let the user try again.
  Invalid command. Please try again.

Rubric

Name	Description	Points
Coding Style	Run cpplint on student code	5
Test Exit	Test user command	5
Test Add	Test user command	20
Test Watch	Test user command	15
Test Delete	Test user command	15
Test History	Test user command	10
Test Recent	Test user command	10
Test Queue	Test user command	10
Test Next	Test user command	10
Total Points		100

Due Dates and Honor

The due date is specified on Blackboard.

This is an ***independent*** programming project, and it is very important that you understand and abide by the ***academic integrity policy*** concerning programming projects. Remember, your personal honor and integrity is far more important than your grade on the project.

Grading

This project is available in GitHub Classroom. Accept the URL on Blackboard and then clone your repository to your machine for development. Your project will be partially graded automatically via GitHub. Please check the grading results each time you check in your code. Your final grade will be based upon your last sync to GitHub before the deadline. I will be manually grading your project as well.

Project Artifacts

The following should be completed by the due date/time specified on Blackboard.

- Check in all source code changes to your GitHub repository. Please check your URL using a web browser to verify that your changes have been synced.
- **Submit the URL for your repository to Blackboard.**