Project LinkedList

Data Structures CSCI 2320

Project Objective

Learn how to use a List ADT in C++ to solve a problem.

Project Overview

Your mission is to implement a **BearFlix** movie list. You will be given a list of movies in a file. You will need to read the file as input and place each item into the list. In addition, you be given a list of movies in a file to add to the existing list. You will need to read the file as input and place each item into the list. Also, you be given a list of movies in a file to remove from the existing list. You will need to read the file as input and remove each item from the list. Finally, you will write out a new file that contains the modified set of movies.

Project Tasks

Review Task

- Review the standard template library list. The interface is similar to the LinkedList we created for the lab.
- Clone the starter repository to your computer.

Visual Studio 2022 Users

- Create your Visual Studio 2022 Project and add the provided `main.cpp` file.
- Copy the following files to the same location as your project/solution files.
  - `mymovies.txt`
  - `add_movies.txt`
  - `del_movies.txt`

Develop your main driver

In your main driver `main.cpp` implement the following:

- Create a list using `std::list`.
- Print the message "Reading in BearFlix movie list..."
- Read in `mymovies.txt`.
- Print the message "Adding new movies..."
- Add each movie in the file to the **end** of your internal list.
- Read in `add_movies.txt`.
- Add each movie in the file to to the **end** of your internal list.

- Print the message "Removing movies..."
- Read in `del_movies.txt`.
- Remove each movie in the file from your internal list.
- As your final step, write all movies in your internal list to a file named `mymovies_updated.txt`. (**hint:** you may need to look for examples of how to extract each item in a list or use the provided `find` method as an example.)
- Print the message "New movie list ready!"

Handle errors in your main driver
Make sure you handle errors according to the following:

- If any of the input files cannot be opened, print "File [filename] cannot be opened." and exit the program using `exit(EXIT_FAILURE)`.
- If you are unable to open the output file, print "File [filename] cannot be opened." and exit the program using `exit(EXIT_FAILURE)`.
- Do not add a duplicate movie to the list. Check to see if the movie is already in the list before adding (**hint:** see the provided `find` method). If the movie is already on the list, print "Movie [movie name] cannot be added to the list." and continue processing.
- Do not attempt to delete a movie not on the list. Check to see if the movie is in the list before attempting to delete (**hint:** see the provided `find` method). If the movie is not on the list, print "Movie [movie name] cannot be removed from list." and continue processing.

Final output

Your final output file must be named `mymovies_updated.txt`. Your main driver output to standard output must be **exactly** like the following:

```
Reading in BearFlix movie list...
Adding new movies...
Movie Decadence and Downfall: The Shah of Iran's Ultimate Party cannot be added to the
list.
Movie Lola cannot be added to the list.
Removing movies...
Movie Zero Day cannot be removed from list.
Movie Sennentuntschi cannot be removed from list.
New movie list ready!
```

Your output file will be verified via GitHub Classroom Actions using different movie lists.

Rubric

Name	Description	Points
AutoGrading Rubric		
AutoTest Setup	Install test code, copy student source, build student and test code	0
Coding Style	Run cpplint on student code	5
Error: Test for missing input files	Compare final output	10
Main Output	Run student main and compare output to test	20
Final Movies List	Compare final movie list to expected output	30
Sub total		65
Manual Grading Rubric		
Verify use of std::list for creation of list		10
Verify use of std::list for adding to list		10
Verify use of std::list for removing from list		10
Verify comment headers		2
Sub total		35
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.