ASSIGNMENT #3c

Purpose

The purpose of this assignment is to practice using pointers, manage memory and work substantially with arrays in an application. You will also be practicing building a User Interface and getting correct input from the user.

Instructions

In this lab you will be creating a TodoList application. The application will allow you to add/edit/delete Todo Items from a Todo List. You will end up with three classes, **`TodoItem`**, **`TodoList`** and **`TodoUI`**.

For Part C (This Assignment) you will be creating a **`TodoUI`** class that will provide the following operations through a user interface:

- Create New Item
- Edit an Item
- Delete an Item
- View All Items
- View Specific Item
- Delete All Items
- Quit Program

For Extra Credit you can add the following:

- Save the list to a file
- Retrieve a previous saved list from a file

To make this all work you will implement a driver (which loads the menus described above) and a UI for the Todo List application. Your driver (i.e. the file containing **main()**) just needs to create an instance of **TodoUI** and then call the **Menu()** function.

You will turn your assignment in by adding/committing/pushing to GitHub

Class - TodoUI

Contains the Menus and all other UI Operations. **All** input and output from this program must happen in the UI

Private Data Members: An instance of **`CinReader`** (or you can create your own input handler if you wish)
A *dynamic* instance of **`TodoList`**

Constructor: Allocates memory for the **`TodoList`** object

Destructor: Frees the memory associated with the dynamic **`TodoList`** object

Member Function 1: Named **`Menu`**. Contains all of the options specified above. All options may be contained in this function, but it would be better to make private member helper functions to help out the **`Menu`** function.

Objectives

- Write a class that will be used by another class
- Use a Private Member Function
- Properly manage memory
- Use pointers and their related syntax
- Create an properly manage a dynamic array and object
- Write proper destructors and use them to manage dynamically allocated arrays
- Implement a User Interface

Requirements

Your code must follow the styling and documenting guidelines presented in class. Please note that I do not give points for style and documentation. You can only lose points. Please make sure your source code is documented correctly and is neatly and consistently formatted using guidelines provided in class.

IF YOUR PROGRAM DOES NOT COMPILE YOU WILL RECEIVE A ZERO!!!

Warnings are treated as non-compile. Use g++ flag –Werror

Deliverables

Commit your files to your GitHub repository. Your complete program should contain the following seven files (and probably `CinReader.h` and `CinReader.cpp`):

- `driver.cpp`
- `todo_item.h`
- `todo_item.cpp`
- `todo_list.h`
- `todo_list.cpp`
- `todo_ui.h`
- `todo_ui.cpp`

Extra Credit Options

In addition to the previously mentioned save to/retrieve from file option above you can add the following functionality to your application:

- Search for and view items using a keyword search on TodoItem text
- Search for and delete items using a keyword search on TodoItem text
- Search for and delete items that have been completed
- Add the following functions
    - Sort by description (case-insensitive)
    - Group by completion status