ASSIGNMENT #3A

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 A (This Assignment) you will be creating a TodoItem Class to represent Todo List Items.

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

Class - TodoItem

Models information about a single item in the Todo List Application

Private Data Members:
A string to contain the description of the item, named **`description_`**
An integer containing the priority of the item (From 1 to 5), named **`priority_`**
A Boolean containing whether or not the item is completed, named **`completed_`**

Constructor:
Has three parameters in the following order and sets those parameters to their corresponding private member variables
String for the description
Integer for the priority, defaults to **`1`**
Boolean for completion status, defaults to **`false`**

Accessors:
An Accessor for each private data member.

Mutators:
A Mutator for each private data member. For **`priority_`** set the priority to **`5`** if an invalid priority is given.

Member Function 1: Named **`ToFile`**. Returns a string containing the description, priority, and completed status, separated by the **`@`** symbol (**`This is the description@3@1`**). Uses the **`scrub`** function to change **`@`** symbols that may be contained in **`description_`**.

Private Member Function 1: Named **`Scrub`**. Has one string parameter. Replaces all **`'@'`** symbols with **`'`` '`** symbols and returns the modified string.

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 and 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.

- **`todo_item.h`**
- **`todo_item.cpp`**