CS 162 Computer Science II

Assignment 3 - Linked Lists 1 (Graded Assignment)

Grade Level A

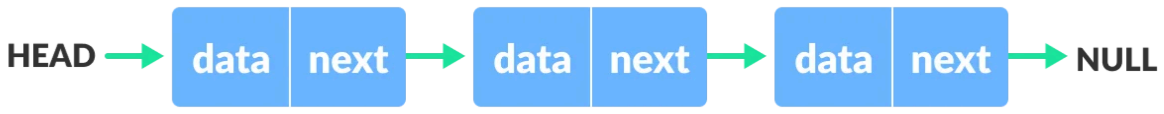
Academic Integrity

You may NOT, under any circumstances, begin a programming assignment by looking for completed code on StackOverflow or Chegg or any such website, which you can claim as your own. Please check out the Student Code of Conduct at PCC.

The only way to learn to code is to do it yourself. The assignments will be built from examples during the lectures, so ask for clarification during class if something seems confusing. If you start with code from another source and just change the variable names or other content to make it look original, you will receive a zero on the assignment.

I may ask you to explain your assignment verbally. If you cannot satisfactorily explain what your code does, and answer questions about why you wrote it in a particular way, then you should also expect a zero.

A linked list is a dynamic data structure that is very easy to update and modify but also is less efficient than an array for retrieving information. Linked lists are the basis of most dynamic data structures, including stacks, queues, and trees (all of which are studied in future CS courses).



Overview

This Homework Assignment includes zyBook exercises and a programming assignment. Please read all of the instructions carefully - there are a lot of details that you need to know!

A task list for a space station, such as the International Space Station (ISS), involves a wide range of tasks and responsibilities to ensure the station's smooth operation and the well-being of the crew. Your task is to keep track of a list of proposed task assignments for a fictional space station and generate some reports from the data.

In this assignment you will create and submit a zip file, **`tasks.zip`** with the following files to the D2L dropbox:

- **`task.h`**
- **`task.cpp`**
- **`tasklist.h`**
- **`tasklist.cpp`**
- **`main.h`**
- **`main.cpp`**
- **`tasks.txt`**
- **`algorithm.pdf`**

Purpose

The focus of this assignment is working with dynamic linked lists and dynamic arrays.

After completing this assignment you will be able to:

- Write code using classes and objects that use dynamic linked lists and arrays
- Use dynamic C-strings
- Use dynamic single link (forward) linked lists
- Insert, unlink, and search for elements in a linked list
- Avoid memory errors such as seg fault and memory leaks
- Organize source code in multiple header and implementation files (.h and .cpp files).
- Implement and use accessors and mutators for a class
- Read from input data files

Task - This is the Grade A version.

What grade would you like?

We understand that your time and money is valuable, and we want to ensure that this programming assignment is tailored to your needs and circumstances. Whether you are a student juggling multiple commitments, a student dealing with personal responsibilities, or a student facing time constraints, or other issues in life, this assignment aims to provide programming skills at different levels that will meet the objectives of this week. With that in mind this assignment has been broken down into 3 different grade levels - Grades A (Exceeds), B (Meets) and C (Approaching). ***This is the Grade A version.***

Tasklist for getting a grade A:

The below set of tasks, if completed correctly per the rubric will get you a grade of **Exceeds** and help master the concepts required to learn the objectives for this week's topic. If you get a lower grade you can resubmit and improve your grade before the one week grace period ends.

zyBook Exercises

You should complete the following zyLabs:

1. zyLab 22.10 - Inventory. This lab introduces the operations of a dynamic linked list.

Once you have completed this zyLab with a score 100% you will need to create a screenshot image of your completion statistics from zyBook, and include that in your algorithm document for this assignment. Note that "successfully completed" means that you received 100% on both labs.

zyLabs are a required element of this assignment **- your submission is not complete and will not receive a grade unless you have completed them and the success statistics are included in your algorithm document.**

Programming Task

For Assignment 1-4, you will work on the same app as the Structs Review Assignment, but the implementation requirements are quite different for each assignment. Please pay extra attention to the section "Some Implementation Requirements" when reading this assignment specification.

You are asked to write a space station app to maintain a list of job assignments/tasks that ensures the station's smooth operation . The app should load all task information from a data file (tasks.txt) once the app is started. It allows the user to view, add, remove, and search for tasks. The app should save the data back to the same data file when the program exits. **In the next**

assignment (Assignment 4, you will add the remove feature and a couple of other requirements).

Please see Sample Run under Criteria for Success.

In order to earn a **A** your Program Should Do

a. Write an interactive text based menu interface (using a loop) that will allow the user to
    i. Enter information for a new task.
    ii. Display information for all the tasks in the database sorted by the person's name with index number for each task.
    iii. Display information for all the tasks in the database with the same Type. **(***new for this grade level***)**
    i. Search for tasks by a certain task name. Use the `strstr` function to do the comparison to do partial search. **(***new for this grade level***)**
    iv. Remove a task by index. **(Don't do this in Assignment 3, just provide an empty implementation. The actual implementation is in Assignment 4).**
    v. Quit

b. For each task, you need to keep track of:
    i. Day of the month (e.g., any number between 1 and 30. For simplicity purposes, let us state that the day of the month goes from 1 to 30.)
    ii. Task name (e.g., Replace ventilation filters)
    iii. Duration in hours (whole numbers)
    iv. Person Name
    v. Type - Operations, Maintenance, Inventory, Communications, or Other - have about 5 different Types of task for the user to choose from.

c. Allow the program to keep looping until the user wants to quit. When the program starts, it should load the tasks from an external file (`"tasks.txt"`) into memory.

d. When it loads the data into the linked list, it should insert them sorted alphabetically by a person's name. ***Do not use a sorting algorithm - as you insert the task into the list, make sure they are inserted in the right position.***

e. When a user enters information about the new task, the program needs to read them in, and insert them in the correct position.

f. When the user quits the program, save them in memory and eventually write them to the external data file (`"tasks.txt"`). The file format could look like:

```
Day;task name;duration;person name;category

5;Replace ventilation filters;2;Robbie Mitchell;1
```

```
28;Replace ventilation filters;2;Carlos Johnston;1
14;Replace ventilation ion filters;2;Karma Thames;1
13;Hull debris sweep;4;Richard Moodyl;0
7;Food system checkup;2;Richard Moody;2
9;Adjust fuel rods;3;Gail Lawhead;0
7;Computer system diagnostic;2;Kelli Weeberly;3
15;Data Transmission and Storage;4;Carlos Johnston;3
```

g. The ';' is used as a delimiter or field separator. Each record ends with a new line character.

h. The numbers 0, 1, 2, 3, and 4 identify the Type that the task belongs to. For example, 0 could be Operations, 1 could be Maintenance, 2 could be Inventory, 3 could be Communications, and 4 could be Others.

Some Implementation Requirements

a. **Before you get started:**
   i. **Check out zyBooks 22.7 Inserting Into A Linked List**
   ii. **Check out the SongType class example in zyBooks 22.8 C++ Example: SongList database using Linked Lists**

b. Use a Class named `Task` to model each task. For the string attributes, such as task name, and person name, you are required to use a **dynamically allocated cstring**. The cstring should be just big enough. For example, the task name attribute should only allocate 17 chars for “`Adjust fuel rods`”.

c. For the day and duration attribute, you can use integer.

d. For the Type attribute, you can use enum or named integer constants.

e. You may not use any while(true) loops or any break statements inside of while loops.

f. No global variables.

g. **Data Validation: (Data validation changes for each grade level - check carefully when you go from grade level to the next.)**
   i. You must do data validation for the menu options (see Sample Run in Criteria for Success).
   ii. You must do data validation for days (must be between 1 and 30 inclusive).
   iii. You must do data validation for hours (must be between 1 and 10 inclusive).
   iv. You must do data validation for the task type (must be between 0 and 4 for Operations, Maintenance, Inventory, Communications, or Other).

    v. Do data validation for all numbers - your program should not crash or go into an infinite loop if I enter a character for a number or invalid data. **(new for this grade level)**

h. Must follow the C++ Style guidelines just like your other assignments.

i. Use a Class named **`TaskList`** to model the collection of tasks. **You are required to use a Linear Linked List of `Task` to implement `TaskList`.** Make sure the **`Task`** is inserted by the person's name. You should not use any sorting algorithms for this. Just insert the **`Task`** at the right position instead.

j. Use `destructor` to deallocate the dynamic memory for the object.

k. Make sure your program is "memory-leak-free" by using valgrind `valgrind --tool=memcheck --leak-check=full executable-file`.

l. You can use the valgrind program on repl.it as well. Watch the video in **Chapter 21. CS162 Dynamic Memory, Debugging Part III.**

m. When using class, please make sure you encapsulate the data which means make all the instance data members `private` and provide accessor methods and mutator methods to access and manipulate the data.

n. For submission, your data file should contain at least 10 lines of test data. It should have test cases for different task names, person names, different tasks for the same day and so on.

o. You must have multiple files for this assignment. **`task.h`** for the Task class and the function prototypes, **`task.cpp`** for the function implementations, **`tasklist.h`** for the TaskList class and the member functions, **`tasklist.cpp`** for the member function implementations, and **`main.h`** and **`main.cpp`** for other functions and implementations, and the main() function. You can have more files if you want.

Criteria for Success

Sample Run

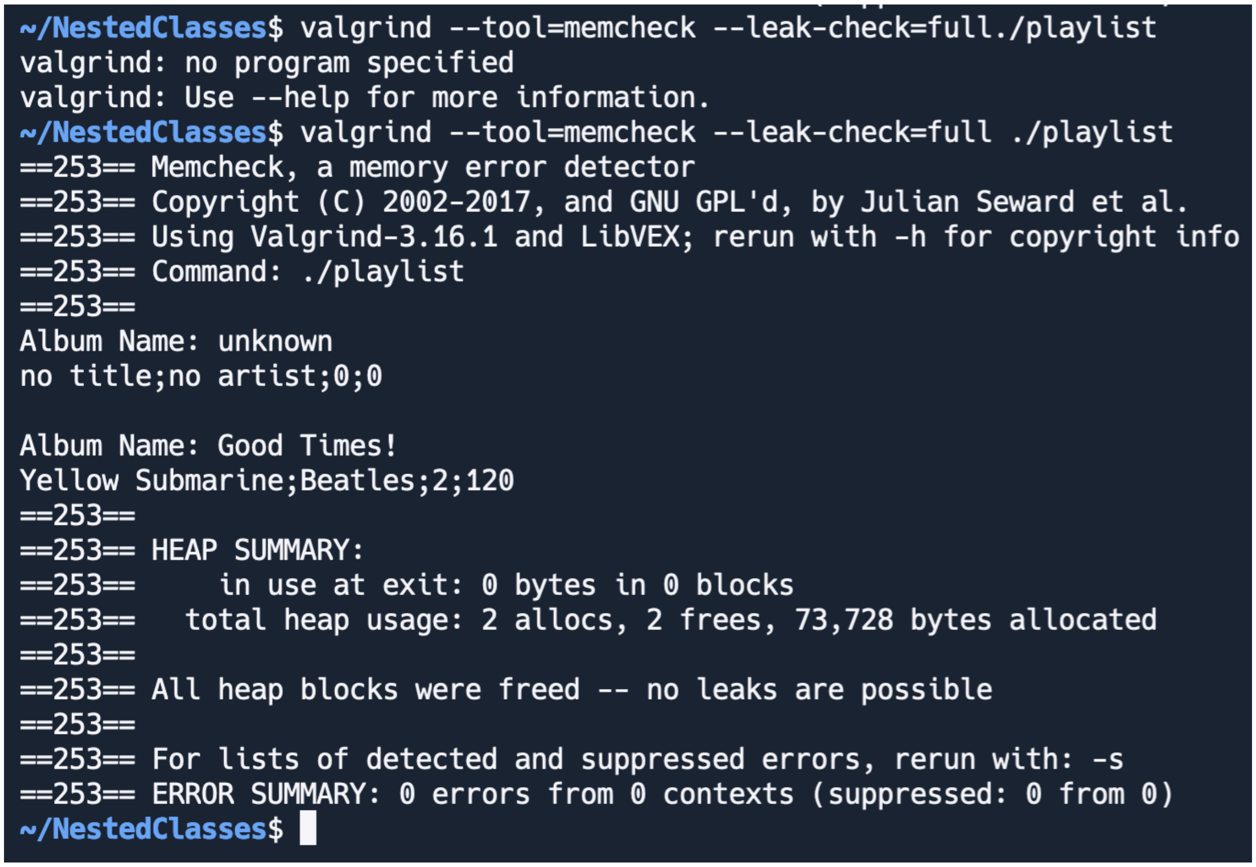
❏ **The below sample run is an example of how the output should look. The user responses are in Blue.**

```
Welcome!
This program will help you manage your tasks for this Space
Station.

Pick an option from below:

(a)Add a new task
(b)List tasks by name
(c)List tasks by Type
```

```
(d)Search by task name
(e)Remove tasks by index
(q)Quit

b
1. 28;Replace ventilation filters;2;Carlos Johnston;Maintenance
2. 15;Data Transmission and Storage;4;Carlos Johnston;Communications
3. 9;Adjust fuel rods;3;Gail Lawhead;Operations
4. 14;Replace ventilation ion filters;2;Karma Thames;Maintenance
5. 7;Computer system diagnostic;2;Kelli Weeberly;Communications
6. 13;Hull debris sweep;4;Richard Moody;Operations
7. 7;Food system checkup;2;Richard Moody;Inventory
8. 5;Replace ventilation filters;2;Robbie Mitchell;Maintenance

Pick an option from below:

(a)Add a new task
(b)List tasks by name
(c)List tasks by Type
(d)Search by task name
(e)Remove tasks by index
(q)Quit

p
Invalid option!! Please try again!

Pick an option from below:

(a)Add a new task
(b)List tasks by name
(c)List tasks by Type
(d)Search by task name
(e)Remove tasks by index
(q)Quit

a
Enter the day of the task (whole numbers between 1 and 30): 78
Invalid day! Must be between 1 and 30 inclusive!
Enter the day of the task (whole numbers between 1 and 30): 8
Enter the task name (50 characters or less): Surface static
discharge
Enter the person’s name (50 characters or less): Steph Kalias
Enter the number of hours (whole numbers between 1 and 10): 3
Enter the task type 0-Operations, 1-Maintenance, 2-Inventory,
3-Communications, and 4-Others): 0

Task added!

1. 28;Replace ventilation filters;2;Carlos Johnston;Maintenance

2. 15;Data Transmission and Storage;4;Carlos Johnston;Communications

3. 9;Adjust fuel rods;3;Gail Lawhead;Operations

4. 14;Replace ventilation ion filters;2;Karma Thames;Maintenance

5. 7;Computer system diagnostic;2;Kelli Weeberly;Communications

6. 13;Hull debris sweep;4;Richard Moody;Operations

7. 7;Food system checkup;2;Richard Moody;Inventory

8. 5;Replace ventilation filters;2;Robbie Mitchell;Maintenance

9. 8;Surface static discharge;Steph Kalias;Operations

Pick an option from below:

(a)Add a new task
(b)List tasks by name
(c)List tasks by Type
(d)Search by task name
(e)Remove tasks by index
(q)Quit

c
Enter Type number(0-Operations, 1-Maintenance, 2-Inventory, 3-Communications, and
4-Others): 0
1. 9;Adjust fuel rods;3;Gail Lawhead;Operations

2. 13;Hull debris sweep;4;Richard Moody;Operations


Pick an option from below:

(a)Add a new task
(b)List tasks by name
```

```
(c)List tasks by Type
(d)Search by task name
(e)Remove tasks by index
(q)Quit

d
Enter task name: ventilation
1. 28;Replace ventilation filters;2;Carlos Johnston;Maintenance

2. 14;Replace ventilation ion filters;2;Karma Thames;Maintenance

3. 5;Replace ventilation filters;2;Robbie Mitchell;Maintenance

Pick an option from below:

(a)Add a new task
(b)List tasks by name
(c)List tasks by Type
(d)Search by task name
(e)Remove tasks by index
(q)Quit

q

Tasks written to file! Thank you for using my program!!
```

- ❑ **Before you get started:**
    - ❑ **Check out zyBooks 22.7 Inserting Into A Linked List**
    - ❑ **Check out the SongType class example in zyBooks 22.8 C++ Example: SongList database using Linked Lists**
- ❑ **Complete zyLabs:**
    - ❑ **22.10 - Inventory - This lab introduces the operations of a dynamic linked list.**
- ❑ Open the Algorithmic Design Document, make a copy, and follow the steps to create your algorithm. Here is a Sample Algorithmic Design Document to model your algorithm.
- ❑ **Be sure to include your screenshot of the zyBooks completion in the algorithm document.**
- ❑ Create your source code files, build and test your program.
- ❑ Code may be written using any development environment, but must use Standard C++.
- ❑ Your program must be free of memory leaks - see image below. You can test your code using valgrind on repl.it or the server.



```
~/NestedClasses$ valgrind --tool=memcheck --leak-check=full./playlist
valgrind: no program specified
valgrind: Use --help for more information.
~/NestedClasses$ valgrind --tool=memcheck --leak-check=full ./playlist
==253== Memcheck, a memory error detector
==253== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==253== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==253== Command: ./playlist
==253==
Album Name: unknown
no title;no artist;0;0

Album Name: Good Times!
Yellow Submarine;Beatles;2;120
==253==
==253== HEAP SUMMARY:
==253==     in use at exit: 0 bytes in 0 blocks
==253==   total heap usage: 2 allocs, 2 frees, 73,728 bytes allocated
==253==
==253== All heap blocks were freed -- no leaks are possible
==253==
==253== For lists of detected and suppressed errors, rerun with: -s
==253== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
~/NestedClasses$ █
```

- ❏ **PSU-bound students are strongly encouraged to do all development on the PCC Linux server.**
- ❏ Please open and compare your work with the grading rubric before submitting.
- ❏ Remember to follow all C++ style guidelines.
- ❏ Download the algorithm document as a PDF file, compress it with the other code files and your txt file, and submit to the D2L dropbox.
- ❏ You must express your algorithm as **pseudocode. Do not use a flowchart in CS 162**. Please note that your pseudocode must follow the syntax requirements shown in the document - **you may not use C++ syntax as a substitute for pseudocode**.

Additional Support

- ❏ Post a question for the instructor in the Ask Questions! area of the Course Lobby.
- ❏ **Check out the SongType class example in zyBooks 22.8 C++ Example: SongList database using Linked Lists**
- ❏ **Check out zyBooks 22.7 Inserting Into A Linked List**
- ❏ **Check out the videos and examples in zyBooks 21. CS 162 Dynamic Variables, Debugging - Part III**
- ❏ Make sure you complete all zybook activities.