CS 162 Computer Science II

Structs Review Assignment (Graded Assignment)

Academic Integrity

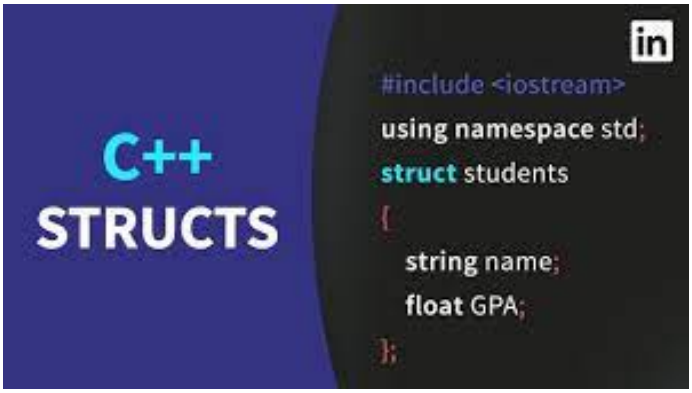
You may NOT, under any circumstances, begin a programming assignment by looking for completed code on StackOverflow or Chegg or any such website, which you can claim as your own. Please check out the Student Code of Conduct at PCC.

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.

In CS 161B the concept of a `struct` is presented in contrast to arrays. Both are aggregate variables that contain more than one value, but while the values in an array all have the same data type, the values in a `struct` may have different data types.

In CS 162 we expand on this notion, and focus on the perspective that a `struct` is a means of creating a user-defined data type. Structs and classes are largely equivalent in C++, and both provide a means of creating new data types for specific purposes.



Purpose

The primary purpose of this assignment is to assess you on the materials covered in 161B. **If you find this assignment challenging, then you must consider taking 161B here at PCC, or work on spending more time to get familiar with these materials to be successful in CS162.**

In this assignment, you will create source code that implements C++ structs.

After completing this assignment you will be able to:

- Write code using struct-defined data types
- Organize source code in multiple header and implementation files (.h and .cpp files).
- Create, initialize, read, update, and display the contents of struct variables
- Use functions with struct parameters and return types
- Read from an input data file
- Use C-strings to represent character strings

Overview

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

One of the great things about living in the Northwest is that we have a great wealth of activities available for us to participate in. We have the waterfront and all that takes place there. We have street fairs and rodeos during the summer. There are farmers markets that will start up soon – always a fun place to go. Or, maybe you are more interested in skiing or snowboarding and like the close proximity to those types of activities?

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

- **`activity.h`**
- **`activity.cpp`**
- **`main.h`**

- **`main.cpp`**
- **`activities.txt`**
- **`algorithm.pdf`**

zyBook Exercises

The first task is to complete zyLabs 15.9 (Student Database). Please note that this exercise assumes that you are familiar with all of the zyBooks material in Chapters 1 through 16 (which were covered in CS 161A and CS 161B), so you might want to review these sections before attempting the zyLabs.

Once you have successfully completed zyLab 15.9 you will need to create a screenshot image of your completion statistics from zyBook, and include that in your algorithm document for this assignment.

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

Introduction

You are asked to write an app to maintain a list of activities that the user likes to participate in. The app should load all activity information from a data file (**`activities.txt`**) once the app is started. It allows the user to view, add, remove, and search for activities. The app should save the data back to the same data file when the program exits. Please see Sample Run under Criteria for Success.

What 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 activity.
   ii. Display information for all the activities in the database sorted by activity name with index number for each activity.
   iii. Display information for all the activities in the database with the same Type.
   iv. Remove an activity by index.
   v. Search for activities by a certain activity name.
   vi. Quit

b. For each activity, you need to keep track of:
    i. Activity name (e.g., Skiing)
    ii. Activity location (e.g., Mt Hood Meadows)
    iii. Activity Level (e.g., Easy, Difficult, Not for the faint of heart etc.)
    iv. Rating (e.g., 1-10 about how fun the experience is)
    v. Type: athletic, food, arts, games, or others - have about 5 different Types of activities 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 (**`"activities.txt"`**) into memory.

d. When it loads the data into the array, it should insert them sorted alphabetically by activity name. Do not use a sorting algorithm - as you insert the activity into the array, make sure they are inserted in the right position.

e. When a user enters information about the new activity, 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 (**`"activities.txt"`**). The file format could look like: (you can use enum or named constants for Type)

```
Skiing;Mt Hood Meadows;Difficult;6;0

Wine Making;Umpqua Valley;Complicated;9;1

Catan;Epic Gaming;Easy;4;3

Oil Painting;Fun Studios;Tricky;6;2

Pottery;Fun Studios;Easy;7;2

Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;0
```

    i. The ';' is used as a delimiter or field separator. Each record ends with a new line character.
    ii. The numbers 0, 1, 2, 3, and 4 identify the Type that the activity belongs to. For example, 0 could be Athletics, 1 could be Food, 2 could be Arts, 3 could be Games, and 4 could be Others.

Some Implementation Requirements

a. **Before you get started:**
    i. **Check out the video store example in zyBooks, 15. CS161B:Section 15.6**
    ii. **Check out the insert sorted into an array example in zyBooks, CS 161B:Section 15.7**

b. Use a struct named **`Activity`** to model each activity. For the string attributes, such as activity name, activity location, and activity level, you are required to use C-string, a

character array terminated by '\0'. You should use the <cstring> library instead of the String class. You can have a maximum of 51 characters for each C-string (50 + 1 for the null character).

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

d. For the Rating attribute, you can use integer.

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

f. No global variables.

g. You must do data validation for any required responses and numbers - for example if you have a set of choices for the Type, then you must validate that the user enters a Type within the choice list. You must validate that the rating is between 1 and 10.

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

i. Use an array of structs to model the collection of activities. You can have a maximum of 30 activities in your list.

j. Use **`strstr`** to do partial comparison to list activities by a location.

k. See the Inventory example in zyBooks. 15: CS161B: Section 15.2 **`(strstr and strcmp examples)`**, the video store example in 15: CS161B: Section 15.6 **`(enums and multiple files and looping menu)`**, and 15:CS161B: Section 15.7, Example to insert sorted into an array. You can find links to these under Criteria for Success.

l. For submission, your data file should contain at least 10 lines of test data. It should have test cases for different activity names, activity locations and the same location with different activity names.

m. You must have multiple files for this assignment. **`activity.h`** for the struct and the function prototypes, **`activity.cpp`** for the function implementations, and **`main.h`** and **`main.cpp`** for other functions and implementations, and the **`main()`** function.

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 activities.

Pick an option from below:

(a)Add a new activity
```

```
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

b
```

1. Catan;Epic Gaming;Easy;4;Games
2. Oil Painting;Fun Studios;Tricky;6;Arts
3. Pottery;Fun Studios;Easy;7;Arts
4. Skiing;Mt Hood Meadows;Difficult;6;Athletics
5. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics
6. Wine Making;Umpqua Valley;Complicated;9;Food

```
Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

p
Invalid option!! Please try again!

Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

d
Enter Type number(0-Athletics, 1-Food, 2-Arts, 3-Games, and
4-Others): 0
```

```
Skiing;Mt Hood Meadows;Difficult;6;Athletics
Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics

Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

C
Enter location name: Hood Meadows

1. Skiing;Mt Hood Meadows;Difficult;6;Athletics
2. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics


Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

e
1. Catan;Epic Gaming;Easy;4;Games
2. Oil Painting;Fun Studios;Tricky;6;Arts
3. Pottery;Fun Studios;Easy;7;Arts
4. Skiing;Mt Hood Meadows;Difficult;6;Athletics
5. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics
6. Wine Making;Umpqua Valley;Complicated;9;Food


Pick the index to remove: 4
```

```
Activity removed!
1. Catan;Epic Gaming;Easy;4;Games
2. Oil Painting;Fun Studios;Tricky;6;Arts
3. Pottery;Fun Studios;Easy;7;Arts
4. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics
5. Wine Making;Umpqua Valley;Complicated;9;Food

Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

a
Enter the activity name (50 characters or less): Rowing
Enter the activity location (50 characters or less): Oaks Amusement
Park
Enter the activity level : Tricky
Enter the activity rating : aaa
Invalid rating! Please enter a valid rating!
Enter the activity rating : 8
Enter Type number(0-Athletics, 1-Food, 2-Arts, 3-Games, and
4-Others): 0

Activity added!
1. Catan;Epic Gaming;Easy;4;Games
2. Oil Painting;Fun Studios;Tricky;6;Arts
3. Pottery;Fun Studios;Easy;7;Arts
4. Rowing;Oaks Amusement Park;Tricky;8;Athletics
5. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics
6. Wine Making;Umpqua Valley;Complicated;9;Food

Pick an option from below:
```

```
(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

f
Enter the activity name (50 characters or less): Snowboarding

Activity found!
```
5. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics

```
Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

f
Enter the activity name (50 characters or less): Skiing
Activity not found!!

Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

q

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

- ❏ **Before you get started:**
  - ❏ **Check out the video store example in zyBooks, CS161B:Section 15.6**
  - ❏ **Check out the insert sorted into an array example in zyBooks, CS 161B:Section 15.7**
- ❏ Complete zyBooks section **15. CS 161B: Structs Part II zyLabs 15.9.**
- ❏ 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++.
- ❏ 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.
- ❏ See the Inventory example in zyBooks. 15. CS161B: Section 15.2, insert sorted into an array example in zyBooks, CS 161B:Section 15.7, and the video store example in 15. CS161B: Section 15.6
- ❏ Make sure you complete all zyBook activities.