CS 3733 Operating Systems assign00

You are required to submit your work through Canvas !!! NO E-MAIL SUBMISSION !!!

!!!! Please carefully check the DUE DATE on Canvas !!!!

!!!! NO LATE SUBMISSION WILL BE ACCEPTED !!!! After the due date, submission link on Canvas may disappear.

The goal of this assignment is to test whether you can write C programs in Linux environment (FOX machines), use Makefile, debug programs (compiling with -g option and using `gdb` and/or `ddd`), check memory leaks (run your program with `valgrind`), zip all your files, and finally submit your zip file through Canvas. Please follow the below directions precisely and submit your work. These are the basic submission steps that you will follow for all the other assignments in this course.
We assume that you have very strong C programming background and has working knowledge of Linux. If you need more help with C programming and data structures, [Click here]Links to an external site. to refresh your knowledge. If you need more help with Linux, please refresh your system programming materials!

For this course, create a directory called `cs3733` on FOX machines at cs.utsa.edu and do all your work under that directory.
For each assignment, create a directory under `cs3733`. For this assignment, call it `abc123-assign0` , where abc123 should be your own abc123!

Part 1:

Download and unzip assign0.zip Download assign0.zipto get all the necessary files and Makefile.
Copy all these files under your abc123-assing0 folder on FOX and run `make` The programs will be compiled. But since most functions are not implemented, you won't get any output when executing `driver1`

First implement a function called `ReadLine()` in `driver1.c` and test it. More explanation about this function and some hints are given in `driver1.c`

Then implement `mylinkedlist` library consisting of `mylinkedlist.h` and `mylinkedlist.c`, which are given as template in the `assign0.zip`.

The function prototypes as well as more explanations are listed in `mylinkedlist.h` BUT you need to implement the exported functions in `mylinkedlist.c` and add additional functions as needed in the next part. Basically, you are asked to develop a simple Linked List library which exports a cell structure and some functions to simplify the creation of a linked list storing an integer, a double, and a string (char *) (e.g., student id, gpa, name). *This is similar to a linked-list assignment you may have completed in CS 2124 Data Structures and will be useful in future assignments!*

Part 2:

Then modify the driver program (`driver1.c`) such that it can interact with the user to get many students' information (id, gpa, and name) and manage the linked list by using the structures and functions from `mylinkedlist` library. Specifically, your driver program will create/initialize a linked list, e.g., `list = NewLinkedList();` and then ask user what to do in a loop as follows:

1 - Create a new student cell with given id, gpa, name info, and add (Enlist) it to the end of the linked list. (if the student id has already been added, don't add that student again, give an error and ask for another option)

2 - Remove (Delist) the first student from linked list and print his/her id, gpa, name info

3 - Print the number of students in the linked list (Length)

4 - Print (id, gpa, name) of a student at a specific index (head of the list is defined as index 0)

5 - Print the list of all students in the linked list. Print (id, gpa, name) of every student

6 - Print the min, average, max GPAs in the linked list

7 - Remove the student with highest GPA and print his/her info (if there are ties, just take the first one you found)

8 - Exit

Enter your choice:

(Depending on choice enter additional info, e.g., id, gpa, name, index)

If the user selects 1, your program should ask user to also enter a student's id (an integer), gpa (a double), name (char *); then create a new student cell with these parameters, e.g., `element = NewStudentCell(id, gpa, name);` and insert `element` at the end of the linked list using `Enlist(list, element);`.

When reading id (an integer) and gpa (a double) you can simply use standard library functions (e.g., scanf("%d", &id). However, when reading name (char *), you need to use the new function (e.g., `name = ReadLine();`) mentioned in Part 1. More information about this function is in driver1.c.

If the user selects 2, your program should remove the first cell from the linked list, e.g., `element = Delist(list);` print the information of `element`, and free it.

If the user selects 3, your program should find the number of students in the linked list, e.g., `n = LinkedListLength(list)` and print n.

If the user selects 4, your program should ask user to also enter an index, then find the cell at that index using `element = GetLinkedListElement(list, index);` print the information of `element`. Note that the element is still in the list, it is not removed. We just look at the information at that index!

If the user selects 5, your program should go over each cell in the linked list, and print the information from that cell.

If the user selects 6, your program should go over each cell in the linked list, and compute the min, average, mag gpa, and print them.

If the user selects 7, your program should search the cell with maximum gpa, then remove it from the linked list, print the information from that cell, and free it.

If the user selects 8, your program should stop the loop. Then free all the cells, linked list, and names. Finally, quit. Use valgrind to make sure there is no memory leakage.

YOU WOULD NEED TO IMPLEMENT NEW FUNCTIONS FOR THE TASKS IN OPTIONS 5, 6, 7, and 8.

Part 3:
Makefile is already created for you to compile your library and the driver program. Look at it and understand how makefile and make works. In future assignments, you are expected to create it by yourself!

Execute your driver1 program. Enter at least 6 students. Select each of other options at least once.
Then, copy/paste your outputs from screen into driver1_output.txt..
Also run your program with valgrind driver1, and make sure there is no memory leakages at the end!

NOTES:
(i) We are not that strict, but when writing your programs, try to follow the programming style guidelines hereLinks to an external site.. But you are expected to include COMMENTS so we can easily understand and follow your logic.
(ii) Always, make sure you release (free) the dynamically allocated memories if you allocate any memory in your programs. So, before submitting your program, run it with `valgrind` to see if there is any memory leakage. Also, if you need to debug your program, compile your program with `-g` option and then run it with `gdb` and/or `ddd`.
(iii) Look at the end of this page for more explanation about some questions that I may get from students.

REPORT:
Create a `REPORT.txt` to answer the following questions:

1. List all of the people that you have collaborated with on this assignment (discussing high-level ideas is OK, but never share implementation details or code). For each person indicate the level of collaboration (small, medium, large). Also write a few sentences describing what was discussed. Indicate whether you were mainly giving help or receiving help.

2. Do you think everything you did is correct? .....

3. If not, give a brief description of what is working and what progress was made on the part that is not working.

4. Comments (e.g., what were the challenges, how to make this assignment more interesting etc.):

SUBMISSION:
Go to `abc123-assign0` and run `make clean`, we don't need your .o or executable files. Just have .h, .c, .txt, and Makefile under assign0.
Then go to `cs3733` and zip -r the whole directory of `abc123-assign0` as a single file and name it as `abc123-assign0.zip`, where `abc123` should be your abc123 ID.
zip -r `abc123-assign0.zip abc123-assign0`
The -r flag stands for "recursive," which tells zip to include all files and subdirectories within the specified folder.

This is very important for TA to get all files at once. So, please follow these directions in future assignments as well!
Please make sure your zip file contains all the files you need to submit! For some reasons, some students submit empty zip files! To avoid the problems later, after uploading your zip files, download them into another folder and unzip to make sure your files are in it!

You need to submit this single zip file through Canvas before the deadline !!! No e-mail submissions !!!

You can submit it multiple times, but we will grade only the most recent submission.

GRADING:
This assignment is worth 30 points (10 points for the implementation of existing functions in library; 10 points for the implementation of the new function that will be added to library; 10 points for the driver program). To receive full credit for this assignment, you must submit it through Canvas before the due date and your submitted zip file must include the source files, input/output/report files and Makefile as described above (.h, .c., .txt. Makefile, outputfiles, REPORT.txt). Please also see the rubric below!