CS 2110

Timed Lab 6

Due Date and Time

Day: Wednesday, November 30th
Time: Before the end of your assigned lab section

Policy

Submission

TURN IN THIS ASSIGNMENT ELECTRONICALLY USING T-SQUARE

SUBMISSIONS WHICH ARE LATE WILL NOT BE ACCEPTED.

EMAIL SUBMISSIONS WILL NOT BE ACCEPTED UNDER ANY CIRCUMSTANCES!

IF YOU FORGET TO HIT THE SUBMIT BUTTON YOU WILL GET A ZERO.

Questions

If you are unsure of what questions mean, the TA's will clarify them to the best of their ability. We will not be able to answer any questions about how to reach a solution to the timed lab questions. You should know how by now!

What's Allowed

- The assignment files
- Your previous homework and lab submissions
- Your mind
- Blank paper for scratch work

What's Not Allowed

- The Internet (except the T-Square Assignment page to submit)
- Any resource on T-Square that is not given in the assignment.
- Textbook or notes on paper or saved on your computer.
- Dropbox (If your hard drive crashes we will let you retake it).
- Email/IM
- Contact in any form with any other person besides TAs

  If you have any questions on what you may not use then assume you can't use it and ask a TA.

Other Restrictions

1. You may not leave the classroom until we have verified that you have submitted the lab. If you leave the classroom without submitting you will receive a zero.
2. **YOU MUST SUBMIT BY THE END OF YOUR LAB PERIOD**. Bear in mind that the clock on your computer may be a few minutes slow. You are supposed to have a full class period to work, and we are letting you use the 10 minutes between classes to make sure you have submitted your work. **WE WILL NOT ACCEPT LATE SUBMISSIONS**, be they 1 second or 1 hour late.
3. The timed lab has been configured to accept one submission. If you accidentally submit or submit the wrong version flag one of the TAs and we will reopen submission for you.

Violations

Failure to follow these rules will be in violation of the Georgia Tech Honor Code **AND YOU WILL RECEIVE A ZERO** and you will be reported to Bill and the Office of Student Integrity.

We take cheating and using of unauthorized resources **VERY SERIOUSLY** and you will be in serious trouble if you are caught.

Remember

1. There is partial credit given, and some of it is just following the directions.
2. We allow you to use your homework assignment.
3. Please don't get stressed out during a timed lab. You have plenty of time; however, use your time effectively
4. Again, remember: Don't get stressed. Partial credit will be given for things you have done correctly. Do the best you can!
5. If you don't know something at least TRY. Do not just walk out of the lab or submit an empty file. Partial credit!
6. Remember what you can and can't use. If you don't know, then don't use it and ask a TA if you can use it. If we catch you with unauthorized resources we will give you a zero, so better to be safe than sorry.

The Assignment

Overview

You will be making a Linked List. It must satisfy the following requirements:

- The list is doubly linked (each node has a *next* pointer pointing to the next node in the linked list and a *previous* pointer pointing to the previous node in the linked list).
- The list has a head and tail pointer stored in the *LIST* struct. These pointers will be *NULL* if the linked list is empty.
- The *next* pointer in the last node and the *previous* pointer in the first node should always be *NULL*.
- The list will contain pointers to various different kinds of data (void pointers).
- You must implement the *create_list, create_node, push_front, pop_back, destroy,* and *to_array* functions.

Requirements

- Your code must not crash, run infinitely, or produce any memory leaks. You will lose a lot of points if these conditions are not met.
- Your code must compile with the Makefile that we have provided! If it does not compile with our Makefile, you will receive a 0.

Running Your Code

We have provided a Makefile for this assignment that will build your project. Your code must compile with our Makefile. Here are the commands you should be using with this makefile:

- To run the tests in test.c: **make run-tests**
- To debug your code using gdb: **make run-gdb**
- To run your code with valgrind: **make run-valgrind**

Note that test cases provided will be extremely similar to the ones used to grade this timed lab. However, you must also check for memory leaks. If your code contains memory leaks, you will lose points.

Deliverables

Submit your **list.c** file on t-square. Make sure it runs with the unmodified header file and Makefile.
Note that if you modify the header file or Makefile you may not receive credit for the assignment!

You may submit only the file listed above. We will not accept any internet links we want the files above and only these files!

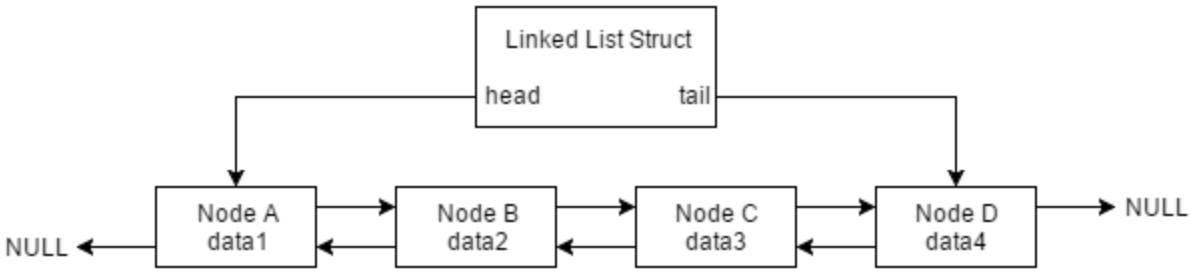
Check over your submission after you submit it. If you submit the wrong file and leave the lab I will not be happy and we will grade what you submit so please check over what you submitted after you submit it!

Have fun and good luck!

Hints are on the next page

Linked List Structure

Here's what the structure of the Linked List should look like:



The Linked List Struct contains a *head* pointer, which points to the first node in the linked list (Node A) and a *tail* pointer, which points to the last node in the linked list (Node D).

Each of the Nodes contains a *next* pointer, which is set to the next node in the linked list, and a *previous* pointer, which is set to the previous pointer in the linked list.

Each of the Nodes has a *data* pointer, which is set when a Node is created. This data pointer can point to any kind of data, so it is of type void pointer.

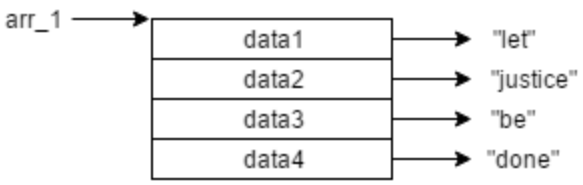
to_array

The *to_array* function should return an array of the data pointers stored in the linked list.

Order matters! The data pointer stored in the head of the linked list should be the first pointer in the array. Likewise, the last pointer in the array should be the tail's data pointer.

Let's say that the values of data1, data2, data3, and data4 are as follows:

```
char* data1 = "let";
char* data2 = "justice";
char* data3 = "be";
char* data4 = "done";
```



Then *to_array* should return a pointer to the beginning of an array of pointers. In this example, that pointer would be *arr_1*. Note: this function should not change what is in the linked list.

Using Malloc

There's a reason this is called the "Dynamic Memory Allocation" Timed Lab. You'll probably need to use malloc and free throughout your code.

Memory Allocation Errors

You should always gracefully handle memory allocation errors, but we won't be testing you on them during this Timed Lab. Assume that *malloc* will never fail and always return a pointer.