CIS*2750
Assignment 0
Deadline: Monday, September 17, 9:00am
Weight: 4%

Description

For this assignment you must expand a doubly-linked list API that has been provided for you. You have been provided with a header file, `LinkedListAPI.h`. Most of the list functionality has already been implemented for you in `LinkedListAPI_incomplete.c`.

This is the first assignment where you must follow a strict specification. As a result, you cannot modify the header file in any way. All your functions must match the header exactly.

Your assignment will be evaluated using an automated test suite, and deviations from the requirements specified in the header and the assignment description will result in severe penalties - up to a zero for this assignment.

You **must** implement **every** function in `LinkedListAPI.h`. If you decide not to implement any of the functions, you **must** provide a stub for them. A stub is a dummy function that does nothing, but matches the signature of the function your are implementing, and returns a dummy value if its return type is not void.

You must provide either a full implementation or a stub for every function listed in `LinkedListAPI.h`, and place it in `LinkedListAPI_incomplete.c`. Rename `LinkedListAPI_incomplete.c` to `LinkedListAPI.c` once you have implemented all the functions listed in `LinkedListAPI.h`.

You need to implement three functions:
- `clearList()`: clears all contents of the list and correctly frees all memory associated with list Nodes and list data. List head and tail are set to NULL and list length is set to 0.
- `createIterator()`: creates `ListIterator` struct and initializes its `current` pointer to the list head. Iterator is returned as a value, to avoid unnecessary mallocs/frees.
- `nextElement()`: returns data associated with the List node that `current` points to, and advances the `current` pointer to the next node in the list.

Additional details on this functions are listed in `LinkedListAPI.h`.

Keep in mind that multiple list functions depend on the functions you need to implement:
- `freeList()` relies on `clearList()`.
- `findElement()` and `toString()` depend on the iterator functionality.

The list API is based on the one typically used in CIS*2520, with a few differences. The biggest one is the addition of the iterator struct and related functions, but there might be other small changes here and there. Please read the header file carefully.

I have provided a sample main file that uses the list API. It must work with your list with absolutely no modifications. This file is meant as a guideline that will let you know whether

you're on the right track. This is not an exhaustive test suite. You must test all of the functions provided by the list API yourself to make sure they match the requirements.

Compile your code on the SoCS server, linux.socs.uoguelph.ca. Use `-Wall` and `-std=c11` flags. We will be using C11 as the default C standard throughout the course. If you want to debug your code using `gdb`, remember to include the `-g` flag.

If you know how to use `valgrind`, make sure you find and fix all memory leaks. For this assignment, you will not lose marks for memory leaks, since we have not yet covered `valgrind` in class. However, starting with Assignment 1, memory leaks will result in deductions.

Evaluation

Your code must compile, run, and have all of the specified functionality implemented.

Any compiler errors will result in the automatic grade of **zero** (0) for the assignment. If you fail to implement all of the required functions - or provide stubs for them - the test harness will fail to compile and you will get the grade of **zero** (0).

Marks will be deducted for:
- Incorrect and missing functionality
- Deviations from the requirements
- Run-time errors
- Compiler warnings
- Failure to follow submission instructions

To make it easier for you to test your code, I have provided a test file: `StructListDemo.c`. Make sure that your list API complies and runs with it. Make sure that you thoroughly test the list.

Make sure you compile and run your code in the lab or on the **linux.socs.uoguelph.ca** server. The course page will have a link to a Linux iso for VirtualBox that matches the SoCS Linux server. If the link isn't up yet, it will be posted shortly.

Submission

Submit your `LinkedListAPI.c` file using Moodle. Do **not** submit `LinkedListAPI.h`. Additional submission guidelines will be added before the due date.

Late submissions: see course outline for late submission policies.

This assignment is individual work and is subject to the University Academic Misconduct Policy. See course outline for details.