MP5: Linked Lists

Objectives:

- Write functions that manipulate linked lists.

Tasks:

1. Write a function that checks whether a linked list is in sorted order.
2. Write a function that checks whether two linked lists are equal.
3. Write a function that removes the minimum element of a linked list.

Task 0: Getting Started

In CLion, create a **New Project**. Make sure it's a "C Executable" project, using the C11 standard. Download the **main.c** file from Moodle into the project directory. (This will replace the **main.c** that was created when you started the project.)

This main.c file contains:

- Definition of a struct for a linked list of integers
- Dummy definitions of the functions that you will need to implement.
- A main function (and a couple of other functions) that serve to test the functions that you will implement.

More typically, we might put the implementation code and the testing code in different files, but I've put it all into one main.c file for convenience. You can build the file and run it. The dummy implementations are written so that they fail most tests. Your job is to replace each dummy implementation with a real implementation.

You should do this one task at a time: Do Task 1, then run the program -- all the Task 1 tests should pass. When that works, move to Task 2 and then to Task 3. Note that you will need to complete Task 2 correctly for the tests for Task 3 to work.

Task 1: Check the order of a list

Implement the function named **checkOrder**. This function takes a single linked list and checks whether the integer values are in sorted order, from low to high. If the list is in order, return 1; otherwise, return 0. If the list is empty, return 0.

Your code does not know and may not depend on the number of items in the list. Your code must work when the input is an empty list.

Compile and run the program. Work on your function until all Task 1 tests pass.

Task 2: Test whether two lists are equal

Implement the function named **equal**. This function takes two linked lists and checks whether both lists contain the same integers, in the same order. If the lists are equal, return 1; otherwise return 0. One or both lists may be empty. (Two empty lists are, in fact, equal to each other.) Note that the lists might have different numbers of elements.

Your code does not know and may not depend on the number of items in the list. Your code must work when one or both of the inputs are empty.

Compile and run the tester. Debug and modify your code until all the Task 2 tests pass.

Task 3: Remove the minimum element of a list

Implement the function named **removeMin**. This function takes a linked list and removes the smallest integer in the list. (NOTE: This is not the first element in this list. This is the element with the smallest value. Remember that integers may be negative!) Return the head of the new, modified list. (This may or may not be the same pointer that was passed in.)

Your code does not know and may not depend on the number of items in the list. Your code must work when the input is an empty list. (When the input list is empty, then of course the resulting list should also be empty.)

Compile and run the tester. Debug and modify your code until all the Task 3 tests pass.

Submission and Grading

Submit your source code to the Moodle MP5 assignment. Do not submit your executable program, just the source code.

Grading:

- 50 pts: checkOrder function passes all tests
- 40 pts: equal function passes all tests
- 10 pts: removeMin function passes all tests

NOTE: We will deduct points if you "rig" the functions to only pass the tests that are given in this code. In other words, if we change the lists created in the main function, your functions should still pass all the tests. During development, you should not even look at any of the code below your functions -- don't tempt yourself to take shortcuts by knowing what the test data will be.