CSE101 Linked List Test Questions: Keeping your data connected

• All code must be written in C/C++.
• Please be careful about using built-in libraries or data structures. The instructions will tell you what is acceptable, and what is not. If you have any doubts, please ask the instructors or TAs.

1 Setting

You are already provided with a linked list code, that handles basic operations. So far, that is "insert", "delete", "print", and "length".

The I/O format has been fixed. The input and output files are given as command line arguments. Each line of the input file is an operation on the linked list. The "print" operation leads to printing of output in the output file.

For the linked list section test, you will have to implement one of the "advanced" functions described below. (These are based on actual interview questions.) *You cannot store the list in an array or some other data structure. You must manipulate the linked list directly to perform these functions.*

1.1 The test questions

To prepare for the test, you should code up various advanced linked list operations. For your test, you will given (some variant of) one of these functions.

• Fancier deletes. The version of delete discussed in class only deletes the first occurrence of a value.

1. Try writing a function that deletes the kth occurrence of a value (for an additional argument k).
2. Delete the last occurrence of a value.
3. Delete all but the first occurrence of a value.

• Testing if list is a palindrome. A palindrome is a list where the first and last elements are the same, the second and second last elements are the same, etc. Solve the classic problem is to test if a linked list is a palindrome (think recursion).

• Rotating a linked list. Think of a linked list as "circular", where the last element wraps around to the head. (Don't actually add such pointers, but just think of it as circular.) Rotation by x means that all elements "move" their position by x, with the end "wrapping around" to the head. Write a function that rotates by x (where x is argument). Write a function that does a "reverse rotate" that rotates backwards by x.

• Reversing a linked list: We discussed reversing a linked list in class. Write a function that reverses the first k elements (where k is an argument). Write a function that reverses the last k elements. For an additional challenge, write a function that reverses every block of k elements (so reverse the first k elements, then reverse the next k elements, etc.). Let's combine reversal with palindromes: convert a linked list to a palindrome by appending the reverse of the list.

1.2 How it works

The linkedlist.h file (that you will get in the test) will already have the right function declarations. Your job is to simply code up the function (that you will only get one as a test question) in the file linkedlist.cpp. Note that if you design more functions to help with your code, you may need to declare them as well in linkedlist.h.

The I/O will be taken care of, so there will be specific operations (in the input file) to perform these functions. So just like we already have "i <INT>" and "d <INT>", I will define more such syntax for all these functions. These will all be explained in a README file. You don't need to deal with any of this, but it might help you debug (if you run on your own inputs).

You will also get a small test input and test output. To make life easier, I will actually give you access to my grading scripts (that you can run directly through a shell script). If it catches a bug in your code, it will give you a test input where your code file. So you will get to see your score *before* you submit.

You will get and can only open the specific Codio box for the test. You cannot open any other Codio box or any other window. If you do so, it will be considered cheating and you will get -10 points.

1.3 What should you do

Umm...how about coding these functions in the linked list Codio box I provided (after I teach linked lists)?

1.4 Other challenging questions

This is a collection of other questions, that might be good practice for exercising your linked list skills. (And great interview practice.)

• Convert a linked list into a circular linked list.

• Sort a linked list of integers in increasing order. (You can do this without changing pointers.)

- Insert an element at the end of the linked list (instead of at the head, as usual).
- Suppose you have a linked list in sorted order. Insert a new element at the correct position in sorted order. (Good coding exercise.)
- Find the middle element of the linked list, making only one pass over the linked list. (It's quite easy if you make two passes. For one pass, there's a cute trick.)
- Get the kth last element of a linked list.
- A classic. Determine if a linked list has a loop. Closely related to previous questions, if you get the trick.