CS 455 Extra Credit Assignment

Fall 2019 [Bono]
Due: Tuesday, Dec 10, 11:59pm
No work accepted after this date.

Introduction and Background

This is an extra credit assignment worth 5%. (So, someone with a perfect score who also got full marks on this could have 105 weighted points at the end, out of 100 points of required work.) It is open to anyone in the class, but is completely optional.

The assignment is to implement some functions that operate on C++ linked lists. It's a little more like a lab, in that you just write those specific functions: we wrote a little interactive test program you should be able to use to thoroughly test your code. However, I would expect it to take more time than a lab session to complete. Unlike a lab you are required to do the work independently (i.e., no partnerships).

Like PA5, this is a multi-file program with a Makefile that uses separate compilation. We did that primarily to give us the freedom to easily test your code using a different test driver than the one we gave you. Hopefully between the Makefile handout (readings for Week 15), the recent lectures, and the last assignment, you are feeling comfortable with multi-file C++ programs.

The assignment files

The files in **`bold`** below are ones you modify and submit. The ones not in bold are ones that you will use, but not modify.

- `ecListFuncs.h` This contains the Node struct definition for our linked lists (but not the method implementations), commented prototypes for the functions you are required to write, and prototypes for a few list utility functions you may want to use (the utility functions are defined in ectest.cpp).
- **`ecListFuncs.cpp`** Implementation file for the the list functions required for this assignment. Stub versions of the functions are already provided so you can compile the program right from the start.
- `ectest.cpp` Test program for your list functions. This is a command-based program, similar to the testlist programs we used in lecture and lab to test linked list code we did there. You probably want to compile and run the program right away to see what it does.
- `Makefile` A file with rules for the "make" command. There are comments at the top of the file telling you how to use it.
- **`README`** See section on Submitting your program for what to put in it. Before you start the assignment please read the following statement which you will be "signing" in the README:

> "I certify that the work submitted for this assignment does not violate USC's student conduct code. In particular, the work is my own, not a collaboration, and does not involve code created by other people, with the exception of the resources explicitly mentioned in the CS 455 Course Syllabus. And I did not share my solution or parts of it with other students in the course."

The assignment

Complete the implementation of the five functions `longestRun`, `removeMultiplesOf3`, `insertMiddle`, and `merge`, described in more detail in `ecListFuncs.h`. Each of the first three will be worth the same amount of credit, and merge will be worth twice as much as the others (i.e., 1 + 1 + 1 + 2). None of the functions depend on

each other, and we will test each of them separately, so you can still get some extra credit by just implementing some of the functions.

The complete specification for these functions (i.e., prototypes + function comments) is given in `ecListFuncs.h`. You put your implementations of these functions in `ecListFuncs.cpp`.

We provided a Makefile; to compile the code use the command:

```
make ectest
```

Grading criteria

Unlike other programming assignments this one will be graded primarily on correctness. As usual, it is up to you to come up with thorough test cases for your code. You should be able to try out all those tests using the test driver provided (`ectest.cpp`).

The other evaluation criteria is not wasting resources. For example, you will lose credit if you implement what should be an O(n) function in $O(n^2)$, or if you have memory leaks, e.g., by doing unnecessary calls to `new`, or not reclaiming memory no longer needed. There are a few additional restrictions on how you must complete `merge` -- see function comments for details.

We will not be evaluating your code on style, except that if your code is so messy/unreadable that we can't evaluate the aspects mentioned in the previous paragraph you will just lose credit for that part. (I would hope things like using good names and consistent indenting are automatic for you by now.) You are not required to write helper functions, but you are welcome to do so to increase code readability and reuse.

Note: you will receive *no* credit for this assignment if the completed versions of your functions print anything.

README file / Submitting your program

Your `README` file must document which of the functions you successfully completed and known bugs in your program (this might be in the form of test cases you tried, but that didn't work), and any special instructions or other information for the grader. It must also contain the signed certification shown near the top of this document.

The submit script will check that the necessary files are present, will compile them, and will make sure there are no cout statements in your code.
