**CS173 Intermediate Computer Science**

**Project 11: List with Linked List**

## OVERVIEW

This is an individual project. You should complete this work entirely on your own. In particular you should not use IDEAS nor CODE from anyone else or from the internet or other non-approved resource. You may get ideas and code from

* The course professor
* Any example code we have done in class
* Anything posted on our canvas page
* The class TA
* Our textbook

For this project, you will implement a List class in c++. We will use the same ADT as our previous List class assignment using dynamic arrays. The List *interface* is exactly the same but the List *implementation* for this assignment will be quite different. We will use a linked list structure to implement our List class.

The most challenging aspect of this project will be to correctly use pointers. Since this List implementation also uses dynamic memory, we will need deep copy versions of all our important methods.

Building a linked list is a rite of passage for cs students. All cs majors build these, and typically in the second course in the major which is an introduction to algorithms and data structures. It is super critical that you are successful in this project. Students should not move forward in the major without being able to successfully implement a linked list. So bring your very best effort and time management skills to this project. I promise that your first assignment in CS271 will be to implement a templated linked-list List class. That is the starting point for CS271 and you will use pointers and list like structures throughout the rest of your Denison CS coursework.

## DYNAMIC DATA

Your list uses linked list structure to hold the actual values. By using a linked list, the list can be made larger if it is full.

Since we are not using an array, we will not have to keep track of a capacity variable whose only purpose was to note the array size. We also will not keep a size variable. Instead, whenever we need to access the size (with the length() method), we will traverse the linked list and count the links to compute the size value.

Since we are doing a deep copy, we will also have to implement the four methods:

* default constructor
* copy constructor
* destructor
* assignment operator

The hardest part of this project is managing all the pointers. You **must** draw a picture with every method that you write. Use the picture and number the steps to transition from the *before* state to the *after* state. You will also have to use the debugger since you will likely encounter exception violations that end the program and don't allow you to easily spot your errors. Drawing pictures and using the debugger are two very important software engineering techniques. Know and expect that part of the learning experience for this project is learning how to implement (pictures) and repair (debugger) your code. The *content* and the *process* are both important learning goals.

## TEMPLATE

Your list class will be a templated class. That means, it will hold a yet-to-be specified datatype. Follow the example template class we created in our CS173 discussions.

You will need to include the List.cpp file inside the List.h file. This is already done for you in the List.h file I have provided. You can no longer compile the List.cpp file separately, because it is not yet a full class until the template type is specified. Use the makefile to compile with "make".

## LOGISTICS

I have given you a complete List.h file. **You should not change the public part of this class**. In the private section, I have given you the structure declaration for the Node and the head pointer. **You should not change or remove these items.**

You may add more private variables and also private methods to List.h.

I have given you a complete listdriver.cpp and a makefile. You may change these files if you would like, but I suggest keeping an original copy somewhere. In particular, you will want to comment out a lot of the list operations until you implement them. Then copy them back in, one at a time, and test each one.

You are to create your own List.cpp file. **You are to submit BOTH your List.h and List.cpp** files as part of this assignment. Your List.h file must have exactly the same public interface – you cannot change any of this. You must submit both files together in the same submission. Do not submit one file in one submission and then the other file in a different submission.

## TESTING

I also suggest you add more extensive testing to your main program. Be sure to think about ways your list class might “break” and then try hard to break it by adding more code to the main program. The listdriver.cpp file I gave you only does cursory testing. I will use a much more extensive test file to see if your code works. Most of the advanced testing comes from testing the dynamic array, looking for memory leaks, and making sure deep copying is working correctly.