CPSC223 Algorithms and Abstract Data Structures
Assignment 1 - Linked List (CPP Review)
Gonzaga University - Spring 2023

Goals:

Implement a simple singly linked list in C++, perform simple benchmarks, practice using pointers, classes, separation of implementation and interface files (.cpp and .h), initialization lists on constructors, mutators, accessors, const data members/functions.

Description:

We have begun our first few classes reviewing some C++ topics, this assignment will serve to help refresh ourselves with writing C++ code. The starter code contains interface files with function definitions and comments describing what needs to be implemented. There is also a Makefile included, and empty .cpp files. Your job is to implement the .cpp files to create a working linked list as described by our .h files.

You must also write some tests to showcase that your code functions properly. These tests should be written inside Main.cpp and each test should be its own function. The number of tests is up to you, however I am asking you to write a minimum of three. More tests means more robust code–focus on testing the edge cases in addition to more general cases to prove your linked list handles them correctly. These can be simple tests with cout statements remarking “PASS” or “FAIL”.

In addition to writing tests, write some code in main to test your findMaxValueFast() and findMaxValueSlow() functions. findMaxValueSlow() should perform a linear search of the linked list, and return the largest value found. findMaxValueFast() does not perform a linear search, but instead you must think of another way to find the largest value within the list – this is the algorithmic task. The simple approach is to search every node, one after another, until we have “looked” at every node value, keep track of the largest we’ve seen, and then return that value. Think about how you might find another way to accomplish this task that is faster than the “simple” approach.

To benchmark your algorithm, use the “time” linux command and use the Wallclock time (this is the ‘real’ time, or ‘total’ time) on a sufficiently large list.

$ time ./LinkedListProg

>>Outputs time metrics of how long this command took to run.

Try to think of a way to make findMaxValueFast() run faster than findMaxValueSlow()

Submission:

-Push your code to your github classroom repo, and create a “release” tag stating it is complete.
-Include a .pdf file that is a writeup of your process for completing this assignment. Include the following things:

- -Instructions on how to run your code
- -A Screenshot of your program passing all your test cases
- -Screenshot(s) of your timed findMaxValueFast and findMaxValueSlow runs
- -A brief explanation of your strategy for completing the findMax algorithms
- -The results of your findMax algorithms
- -One paragraph reflection on the following:
  - -what was easy
  - -what was hard
  - -anything you would do differently next time

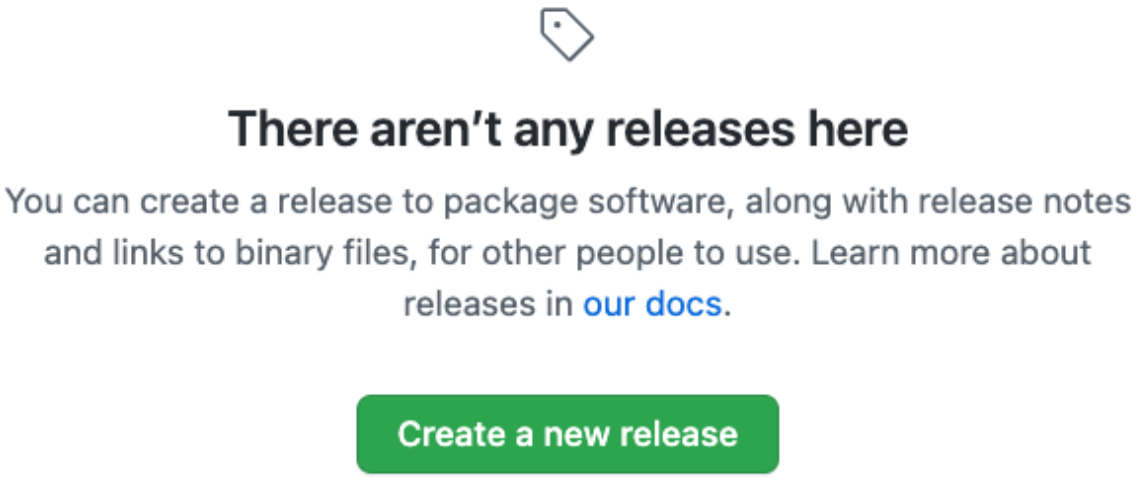
Grading Rubric:

Node implementation ● Constructor …2pts ● Parameterized Constructor …5pts ● getValue accessor …2pts
Linked List implementation ● Constructor …3pts ● Parameterized Constructor …4pts ● removeByIndex …5pts ● removeByValue …5pts ● insertValue …5pts ● findValue …5pts ● getValue …5pts ● addToFront …5pts ● addToEnd …5pts ● getSize …3pts ● printList …5pts ● getMaxValueSlow …5pts ● getMaxValueFast …10pts
Test Cases …9pts ● Wrote at least 3 test cases that all pass
Program Compiles and runs correctly …10pts ● Write cout statements indicating each function is being called and printing the results of each function call. ● When you type make into the terminal it should compile and run, and output information to the screen that clearly shows each function is working properly.
Proper comments and style …7pts
Total: 100pts (**Non-compiling code can receive a maximum of 10 points)**

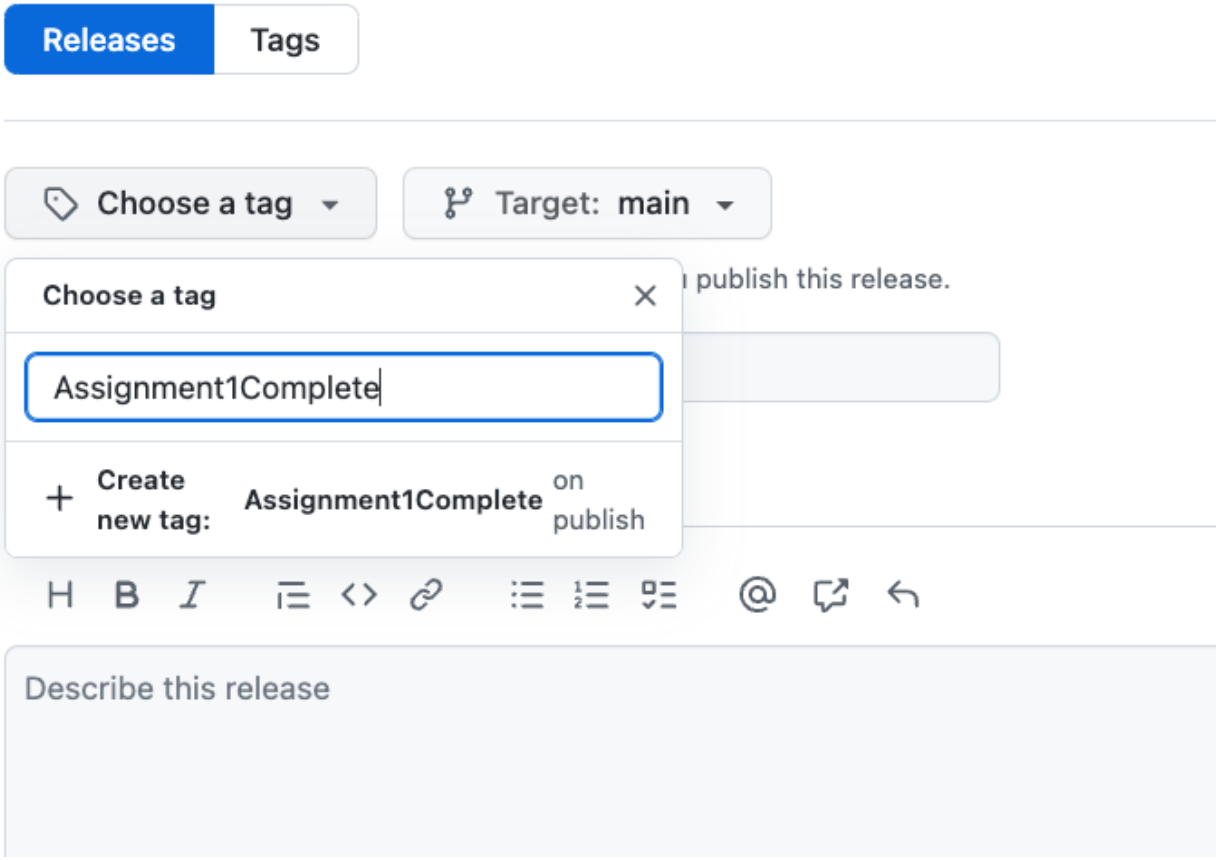
Github Instructions:

To create a release, click on “tags”

Then on “Create a new release”



Then click “Choose a tag” and you must enter some text to create a new tag:



After typing in the name of your release, click the “+ Create new tag:” and publish the release.