Santa Monica College
CS 20A, Fall 2014

Programming Project 2 - Due: Sunday, October 19 at 11:59 PM

You kind of liked learning about linked lists in class but you still have your doubts those silly looking box and arrow diagrams on the board actually work. You decide to try some of them out and see for yourself.

Part 1:

You will implement a simplified singly linked list class. The SimpleList.h header file has been provided to you together with an incomplete SimpleList.cpp file. Please provide the implementation for the missing methods.

Hints:

Start with the simpler methods and do not move on to the next method unless you have tested and are absolutely sure the method you just completed is 100% correct. Bugs are much harder to find when they accumulate.

It is okay to move to Part 2 before you have completed the get(i) and set(i) implementations as Part 2 does not require them. You can come back to fill these in later. Make sure that get(i) works for i=0 though, since Part 2 will need that.

You do not have to worry about the client calling get(i)/set(i) for i>=size. You can assume the client is well behaved.

Part 2:

Whew! The core of Part 1 has been completed and tested. This mambo jumbo actually works! (You have already forgotten about the few little bugs that almost drove you crazy; It was all worth it.)

With the SimpleList class on hand you decide to experiment using it as the basis for implementing a stack. The SimpleStack.h header file has been provided to you. Please provide the implementation in SimpleStack.cpp.

Hints:

If you find yourself writing more than **1-2 lines of code** for each of the methods you implement, you are likely doing something wrong.

You do not have to worry about the client calling peek() or pop() on an empty stack. You can assume the client will always call isEmpty before to make sure it is not empty.

Part 3:

The SimpleStack class has a method that prints the list elements in order but you are wondering about how you could print them in reverse order. You could use the get(i) function from back to front, but you decide to be clever. You will use the stack class from Part 2 instead. This is your plan:

- get each of the list elements in order and push them one by one to the stack
- now that all the elements are in the stack, while the stack has elements left in it, you will print the top of the stack and pop

Provide this implementation for the empty bodied function in CS20AProj2.cpp.
Write a **brief (5 lines max)** analysis (as a comment above the function) of the performance of this function and give the big O for it. Make sure your analysis includes information about which part(s) of the function lead to the O value.

Deliverables:

You should submit a zip file named project2_first_last.zip (where first and last are your first and last name) containing **ONLY the files below.**

SimpleList.cpp
SimpleStack.cpp
CS20AProj2.cpp

How you get points:

– Part 1	65 points
– Part 2	20 points
– Part 3	15 points

How you lose points:

- You do not follow the given directions and decide to make changes "for fun". Specifically, **do not change the header files given to you**, or the .cpp files you submit will not work for me.
- You use arrays or any array based data structure of any kind.
- You submit your whole workspace or executable files. **Submit only the files the project asks for.**
- If your solution is grossly inefficient. Your solution should be **efficient to a level seen in class for similar problems**.
- If any of your code prints anything at all on the console except for the printList and printListReversed functions. **Remove all your print outs, debug statements, etc. Clean up your code and do not leave clutter behind.**
- Your code has no comments where needed. **Comment your code appropriately.**