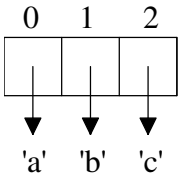
Data Structures (810:052) Lab 4 Name:_____

Objective: To experiment with stack implementations in Python to obverse the importance of big-oh analysis of operations. To gain experience implementing a Stack ADT (Abstract Data Type) using a Python list, Lambert's Array class, and a linked-list implementation.

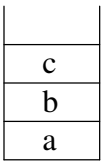
To start: Download and extract the file lab4.zip from:
http://www.cs.uni.edu/~fienup/cs052f10/labs/index.htm

Part A: Recall in yesterday's class that I conjectured (after reading on the web) that Python's built-in list is implemented as an array of references (pointers) to list items. For example, the list ['a', 'b', 'c'] would really be implemented as the array.



This has performance ramifications for possible Stack implementations using a list representation. The two possible list representation of a Stack being:

"Abstract" Stack



I. Stack with Bottom at list [0] — 0 1 2 ['a', 'b', 'c'] — Files in lab4.zip (stackClass.py)

II. Stack with Top at list [0] — 0 1 2 ['c', 'b', 'a'] — (stackClassAlt.py)

Complete the expected big-oh notation for each Stack implementation assuming "n" items in the stack.

	push(item)	pop( )
Stack with Bottom at list [0]		
Stack with Top at list [0]		

Use the file `testList.py` to time the pushing of 300,000 items onto a stack and then popping them off. Complete the following timing table:

	Time (seconds) to push 300,000 items	Time (seconds) to pop 300,000 items
Stack with Bottom at list [0]		
Stack with Top at list [0]		

a) Explain why one implementation is so much slower that the other.

b) Why do you suppose it takes less time to pop than push the same number of elements?

After you have answered this questions, raise your hand and explain them.

Part B: Section 13.2 discussed the Array class for implementing data structures, and section 14.4's ArrayStack class (in file `stack.py`) is an Array implementation of a stack. Starting with the `testList.py` program of part A, make a new `testArray.py` program that times the pushing of 300,000 items followed by the popping of 300,000 items.

a) With-respect-to timing, predict which list implementation of Part A, closest compares to the ArrayStack implementation.

b) What is the timing for the `testArray.py` to push 300,000 items?

c) What is the timing for the `testArray.py` to pop 300,000 items?

d) Why do your `testArray.py` implementation differ from the closed list implementation you predicted in (a)?

After you have answered the above questions, raise your hand and explain your answers. Nothing needs to turned in for this lab.

If you complete all parts of the lab, nothing needs to be turned in for this lab. If you do not get done today, then show me the completed lab in next week's lab period. When done, remember to log off.