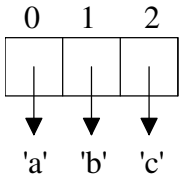
Data Structures (810:052) Lab 5 Name:_____

Objective: To experiment with stack implementations in Python to observe 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, ArrayStack class, LinkedStack class, and LinkedQueue class.

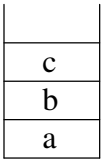
To start: Download and extract the file lab5.zip from:
http://www.cs.uni.edu/~fienup/cs052s11/labs/index.htm

Part A: Recall 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 (NOT Lamert's ArrayStack class). The two possible list representation of a Stack being:

"Abstract" Stack



		Files in lab4.zip
I. Stack with Bottom at list [0]	0 1 2 ['a', 'b', 'c']	(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 discussed the ArrayStack class and LinkedStack implementation of a stack (both in file `stack.py`). The `testArrayAndLinked.py` program times the pushing of 300,000 items followed by the popping of 300,000 items using an ArrayStack and a LinkedStack objects.

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

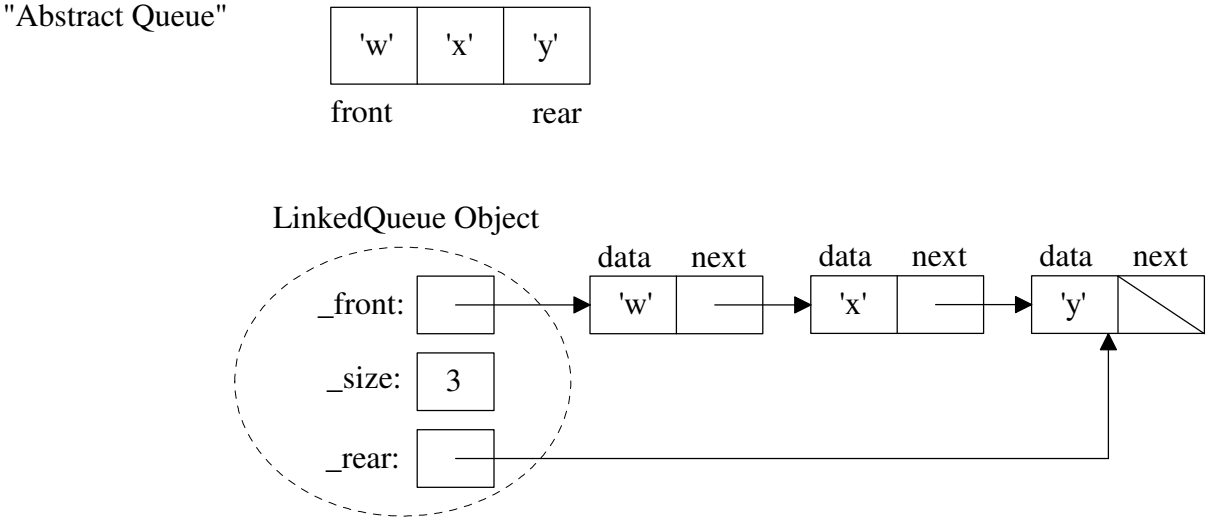
b) Use the file `testArrayAndLinked.py` to time the pushing of 300,000 items onto a stack and then popping them off. Complete the following timing table:

	Time (seconds) to push 300,000 items	Time (seconds) to pop 300,000 items
ArrayStack		
LinkedStack		

c) Why do your `ArrayStack` times differ from the closed list implementation you predicted in (a)?

After you have answered the above questions, raise your hand and explain your answers.

Part C: The `Node` class defined in node.py is used to dynamically create storage for a new item added to a singly-linked list implementation of the `LinkedQueue` class in file `queue.py`. Conceptually, a `LinkedQueue` object would look like:



The file `queue.py` contains the start of the `LinkedQueue` class. Complete the `LinkedQueue` class, and thoroughly test it using the menu-driven program in the `testQueue.py` file.

After thoroughly testing your circular-array implementation, raise you hand and demonstrate your queue.

If you complete all parts of the lab, nothing needs to be turned in for this lab. If you do not get done today, next lab period will be a "catch-up" session. When done, remember to log off and take your USB drive.