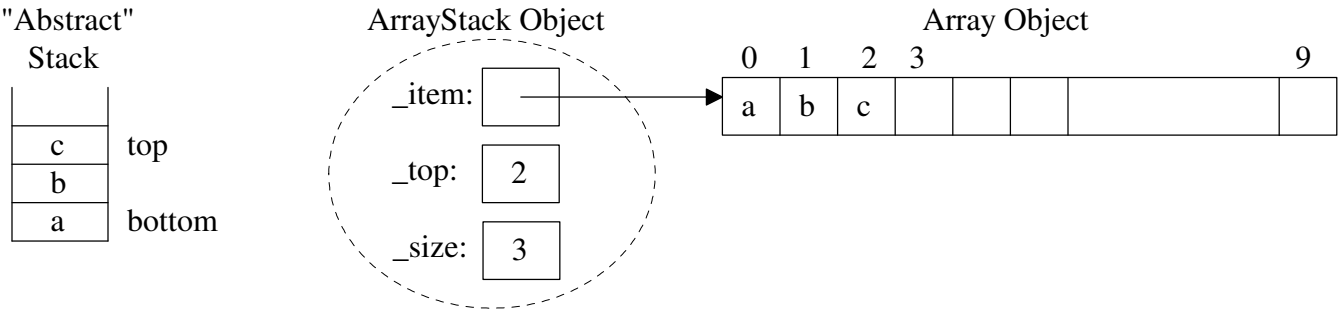
1. The ArrayStack class (section 14.4) is an Array implementation of a stack.

:



a) Complete the big-oh notation for each stack methods assuming the Array implementation: ("n" is the # items)

	__init__ (constructor)	push(item)	pop( )	peek( )	len( )	isEmpty( )
Big-oh						

b) The below `push` method of the ArrayStack class doubles the array size when it fills. For a stack of size n, how many moves due to resizing of arrays occurred?

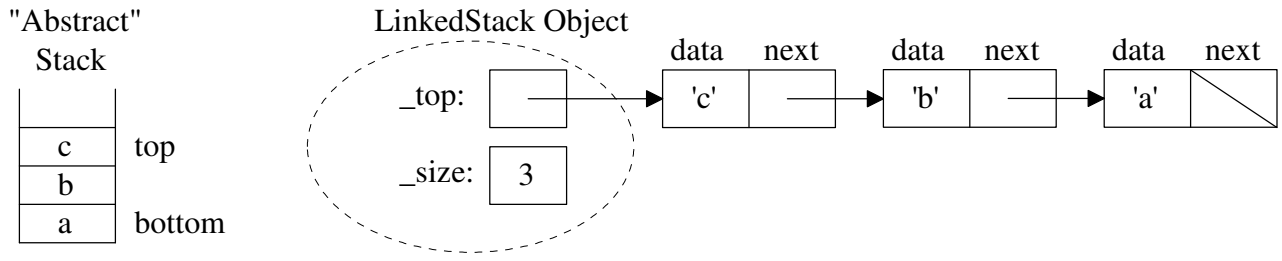
```
def push(self, newItem):
    """Inserts newItem at top of stack."""
    # Resize array if necessary
    if len(self) == len(self._items):
        temp = Array(2 * self._size)
        for i in xrange(len(self)):
            temp[i] = self._items[i]
        self._items = temp
    # newItem goes at logical end of array
    self._top += 1
    self._size += 1
    self._items[self._top] = newItem
```

c) Modify the `push` method of the `ArrayStack` class in the file `stack.py` such that it reduces the array size when the logical size of the array is less than or equal to one-fourth of its physical size and its physical size is greater than the default capacity. In this case, reduce the physical size of the array either to half its physical size or to its default capacity, whichever is greater.

```
def pop(self):
 """Removes and returns the item at top of
    the stack.
    Precondition: the stack is not empty."""
 oldItem = self._items[self._top]
 self._top -= 1
 self._size -= 1
 # Resizing the array if necessary









 return oldItem
```

```
"""  File: node.py   Node class for one-way linked structures.  """
class Node(object):
    def __init__(self, data, next = None):
        """Instantiates a Node with default next of None"""
        self.data = data
        self.next = next
```

2. The `Node` class (in `node.py`) is used to dynamically create storage for a new item added to the stack. The `LinkedStack` class (in `stack.py`) uses this Node class. Conceptually, a `LinkedStack` object would look like:




```
"""  File: stack.py   Stack Implementations """

from node import Node

class LinkedStack(object):
    """ Link-based stack implementation."""

    def __init__(self):
        self._top = None
        self._size = 0

    def push(self, newItem):
        """Inserts newItem at top of stack."""



    def pop(self):
        """Removes and returns the item at top of the stack.
        Precondition: the stack is not empty."""









    def peek(self):
        """Returns the item at top of the stack.
        Precondition: the stack is not empty."""
        return self._top.data

    def __len__(self):
        """Returns the number of items in the stack."""
        return self._size

    def isEmpty(self):
        return len(self) == 0

    def __str__(self):
        """Items strung from bottom to top."""

        # Helper recurses to end of nodes
        def strHelper(probe):
            if probe is None:
                return ""
            else:
                return strHelper(probe.next) + str(probe.data) + " "

        return strHelper(self._top)
```

a) Complete the `push` and `pop` methods.

b) Trace the `__str__` call for the above stack.

c) Complete the big-oh notation for each of the following stack methods assuming this linked implementation:

	__init__	push(item)	pop( )	peek( )	len( )	isEmpty( )
Big-oh						