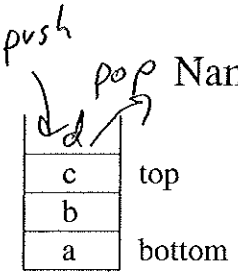
Data Structures (CS 1520) Lecture 4 Name: Mark F.

1. An "abstract" view of the stack:



Using an array implementation would look something like:

n = # items in stack

top: ~~2~~ 3 max: 100

Complete the big-oh notation for the following stack methods assuming an array implementation: ("n" is the # items)

	push(item)	pop( )	peek( )	size( )	isEmpty( )	isFull( )	Constructor
Big-oh	O(1)	O(1)	O(1)	O(1)	O(1)	O(1)	O(1)

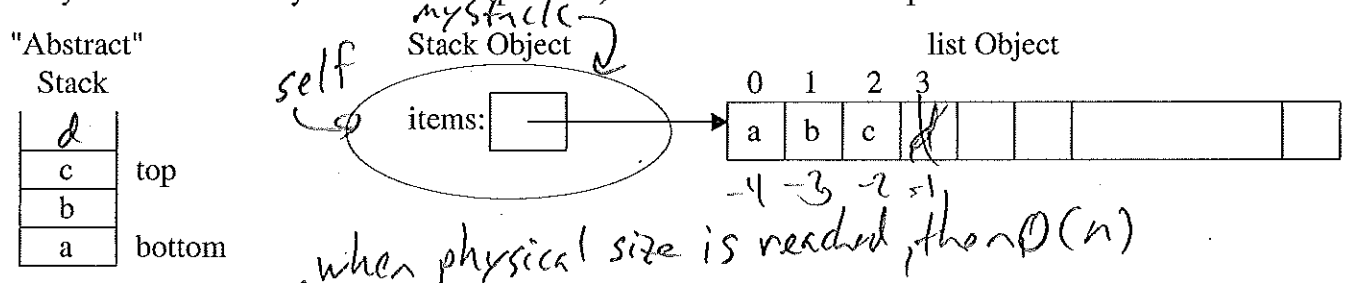
2. Since Python does not have a (directly accessable) built-in array, we can use a list.

```
class Stack:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop()

    def peek(self):
        return self.items[len(self.items)-1]

    def size(self):
        return len(self.items)
```

myStack = Stack()

return len(self.items) == 0

Since Python uses an array of references (pointers) to list items in their implementation of a list.



when physical size is needed, then O(n)

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

	push(item)	pop( )	peek( )	size( )	isEmpty( )	__init__
Big-oh	O(1)*	O(1)	O(1)	O(1)	O(1)	O(1)

b) Which operations should have what preconditions?

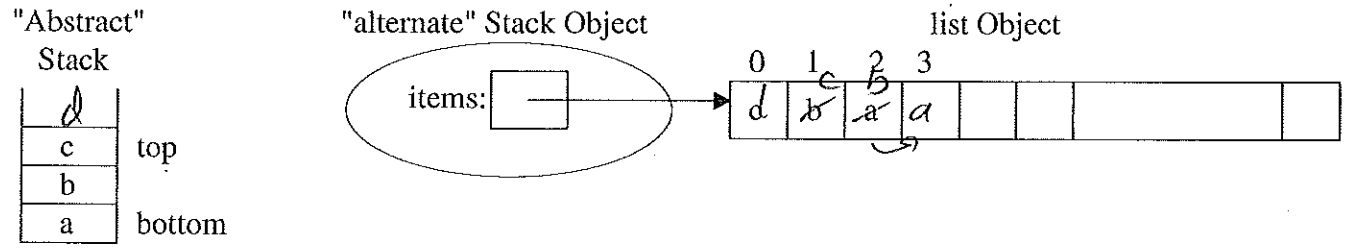
pop - stack is not empty

peek - " " " "

3. The text's alternative stack implementation also using a Python list is:

```
class Stack:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def push(self, item):
        self.items.insert(0,item)

    def pop(self):
        return self.items.pop(0)

    def peek(self):
        return self.items[0]

    def size(self):
        return len(self.items)
```

Since an array is used to implement a Python list, the alternate Stack implementation using a list:



a) Complete the big-oh notation for the "alternate" Stack methods: ("n" is the # items)

	push(item)	pop( )	peek( )	size( )	isEmpty( )	__init__
Big-oh	O(n)	O(n)	O(1)	O(1)	O(1)	O(1)

4. How could we use a stack to check if a word is a palindrome (e.g., "radar," toot)?

```
def isPal(word):
    myStack = Stack()
    for index in range(0, len(word)//2):
        myStack.push(word[index])
    palFlag = True
    for index in range((len(word)+1)//2, len(word)):
        if word[index] != myStack.pop():
            return False # palFlag = False
    return True
    return palFlag
```

5. How could we check to see if we have a balanced string of nested symbols? ("((([])){()}[])")

empty stack
Scan string from left-to-right.
If opening symbol, then push on stack
else if closing symbol, then pop stack and
check for matching opening symbol