CSC/CPE 202: Spring 2020

LAB-3 (Due 4/29)

Note: Always your Name and Section must be as header of your files.
If you are collaborating with anyone or any sources, please write the name of your collaborator(s) in the header of the file.

Demo your work before deadline ends and submit your .py files. Make sure follow the design recipe. No need for boilerplate and template.

Goal:
The goal of this lab is to implement the Stack Abstract Data Type using two different implementations:

1) **Array implementation (using the built in List construct in Python)**
2) **Single Linked List discussed in the class**

As part of the Stack definition as we discussed in the class for this lab, we will specify a capacity for the stack. In the case of the Python List implementation, you will allocate a list of size capacity and use this to store the items in the stack. Since Lists in Python expand when more storage is needed, you must avoid using any functions that would cause a List to expand (see the list of forbidden methods/operations below). This prevents the stack from growing if the user accesses the List through the given interface. (This prevents the stack from using more space than a user might want. Think of this as a requirement for an application on a small device that has very limited storage.) For consistency, in the simple linked data structure implementation, we will also have a capacity attribute for the stack. In this case, when a user attempts to push an item to a full stack, your push function (method) should raise an IndexError exception. Similarly, if a user tries to pop from an empty stack, your pop function (method) should raise an IndexError exception.

You are NOT allowed to use the following Python List operations (we will check!):

all built-in list methods including:

- append()
- insert()
- extend()
- remove()
- pop()
- == and != between lists
- del
- in
- len
- + (concatenation of lists)
- List slicing (e.g. some_list[2:9])

Additional Requirements:

All stack operations must have O(1) performance.
Your stack implementations must be able to hold values of None as valid data

The following starter files are available on Canvas, or at these links:

1) stack_array.py: Contains an array (Python List) based implementation of the StackArray class
2) stack_linked.py: Contains a linked based implementation of the StackLinked class
3) stack_tests.py: Contains your set of thorough tests to ensure your implementations work correctly. These tests must run correctly on ANY implementation that follows the specification.

Submissions:

1) **stack_array.py** Contains an array based implementation of the stack class. The class must be called: **StackArray** .
2) **stack_linked.py**: Contains a linked list based implementation of the **stack** class. The class name must be **StackLinked.**
   Both implementations must follow the above specification and be thoroughly tested.
3) **stack_tests.py** contains your set of test cases for both implementations to ensure your classes work correctly. You need to have at least two test cases for each function (or method) except one test for size

Rubric:

1) stack_array.py	(40 points)
a. six functions	15
b. design recipe	10 (data definition, purpose statement and signature)
c. test cases	15 (at least 3 cases for each function)
2) stack_linked.py	(40 points)
a. six functions	15
b. design recipe	10 (data definition, purpose statement and signature)
c. test cases	15 (at least 3 cases for each function)
3) Extra test cases	20 points
	100 points