Project 1: Stack Implementation

Goal: The goal of this project is to implement the Stack Abstract Data Type using the built in List construct in Python and the Stack ADT using the simple linked list data structure that will cover in class.

As discussed in class you allocate a list of size stack_capacity and use this to store the items in the stack. Since Lists in python expand when more storage is needed you will have to provide a mechanism to prevent the list *inside* your stack from growing. This really prevents the stack from growing if the user only accesses it through the given interface but that is fine for the purposes of this exercise. (This prevents the stack from using more space that a user might want. Think of this as a requirement for an application on a small device that has very limited storage.) 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.

In the PolyLearn there are **starter files: 1) stack_array.py and 2)stack_linked.py** **stack_array.py** provides the following as a starting point**s**:

stack_array.py

```
class StackArray:
    """Implements an efficient last-in-first-out Abstract Data Type using a Python List"""

    def __init__(self, capacity):
        """Creates and empty stack with a capacity"""
        self.capacity = capacity              # Capacity of your stack
        self.items = [None]*capacity          # initializing the stack
        self.num_items = 0                    # number of elements in the stack

    def is_empty(self):
        """Returns true if the stack is empty and false otherwise"""

    def is_full(self):
        """Returns true if the stack is full and false otherwise"""

    def push(self, item):

    def pop(self):
        """Returns item that is popped from stack"""

    def peek(self):

    def size(self):
       """Returns the number of elements currently in the stack(not capacity)"""
```

Due Date: 10/11 @11:55PM for 100%

10/12 @11:55PM for 80%

Submit to PolyLearn three files:

- **stack_array.py** Contains an array based implementation of the stack class. The class must be called: **StackArray** .
- **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.
- **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).

You are NOT allowed to use the following Python List operations:

- append()
- insert()
- extend()
- remove()
- pop()
- del()
- + (concatenations)
- List slicing

Make sure that you follow the design recipe. (No need for template, no need for boilerplates)

Note: Your class names, function (or method) names, and file names must follow the spec of the project.