PA - 1
Playing with Stacks!

Description:

For this programming assignment, you will implement a **Stack ADT** whose **size can grow** as elements are inserted into the stack. You will **build four different implementations** of the Stack ADT. The **first two implementations** will be **array-based** (using NumPy arrays in Python). Each of these two should take an **initial capacity** for the stack in the **constructor** (the capacity attribute should be updated as and when the stack is extended). The only difference between these implementations will be what happens when the Stack is full.

- For the **ArrayStack** implementation, you should increase the size of the array by a **constant amount**.
- For the **DoublingArrayStack** implementation, you should **double the size** of the array.
- For the **LinkedListStack** implementation, you will build a Stack using **Linked List**.
- For comparison, you will also include a fourth implementation that uses the **deque structure from the collections module** in Python to see a pre-built optimized implementation.

You will then write code for the function **infix_to_postfix** which will take in an infix expression to be converted to postfix form as discussed during the lecture and a stack object (instantiated from one of the classes above) that will be used to manage the operators.

Coding Portion (30 Points):

- You will be using the concepts of Object-Oriented Programming to execute the implementations through classes for each of the versions (except the fourth).
- Start with the following template: *Stack.ipynb*, and fill in all of the member functions. Use the **ArrayStack** class as a template for both **ArrayStack** and **DoublingArrayStack.**
- For the **Linked List** implementation, you will need to create another class for the individual nodes in the Linked List.
- Be sure to test the correctness of your algorithms and implementations. Ensure all stack operations of stack ADT (push, pop, top, size, isEmpty etc.) and infix_to_postfix() are implemented and tested to be working as expected.
- Your code will be graded based on whether it produces correct output on test cases.

Submission:

Once you have completed the assignment, you should upload the python script to the **PA-1** codePost portal. Please ensure the following while submitting:

- Once satisfied with your code, you should download the file as a python script (.py file), by going to **File** > **Download** > **Download .py**
- The name of the file should be **PA1.py**
- Upload the python script file to codePost under the PA-1 assignment.
- You can run the test cases on your script up to a limit of 50 times.
- Once satisfied with the test runs, complete your submission.