Project 1 – Abstract Data Types (ADT)

The goal of this part of the project is to implement the Stack and Queue ADTs using the built in List construct in Python and the Stack and Queue ADTs using the simple linked list data structure covered in class.

Stack Implementation

You are to 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 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 than a user might want. Think of this as a requirement for an application on a small device that has very limited storage.) In the case when a user attempts to push an item on to a full stack, your push function (method) should raise an IndexError. Similarly, if a user tries to pop an empty stack, your pop function (method) should raise an IndexError.

Provide two implementations: one using Python lists, and one using the linked list structure.

Download the starter files stack_array.py and stack_linked_list which provides a starting point for implementation.

Demonstrate the following files:

- **stack_array.py** and **stack_linked_list.py** containing a list-based implementation of stack and a linked list implementation of stack, respectively. The classes must be called: **StackArray** and **StackLinkedList** . Both implementations should follow the above specification and be thoroughly tested.
- **test_stack_array.py** and **test_stack_linked_list.py** contain your set of tests to ensure you classes work correctly

Note: Your class names, function names and files name must follow the spec of the project.

Queue Implementation

You are to allocate a list of size 'capacity' and use this to store the items in the queue. Since Lists in python expand when more storage is needed you will have to provide a mechanism to prevent the list *inside* your queue from growing. This really prevents the queue 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 queue 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.) In the case when a user attempts to enqueue an item in to a full queue, your enqueue function (method) should raise an IndexError. Similarly, if a user tries to dequeue an empty queue, your dequeue function (method) should raise an IndexError.

Provide two implementations: one using Python lists, and one using the linked list structure.

Download the starter files **queue_array.py** and **queue_linked_list** which provides a starting point for implementation.

Demonstrate the following files:

- **queue_arrat.py** and **queue_linked_list.py** containing a list-based implementation of stack and a linked list implementation of stack, respectively. The classes must be called: **QueueArray** and **QueueLinkedList** . Both implementations should follow the above specification and be thoroughly tested.
- **test_queue_array.py** and **test_queue_linked_list.py** contain your set of tests to ensure you classes work correctly

Note: Your class names, function names and files name must follow the spec of the project.