LE - 3
Queue ADT implementation using Arrays

Introduction:

This exercise is a hands-on component for the concepts acquired in the lecture on Queue ADT. This introduces you to the Abstraction concept and how a data structure and its operations can be created according to the requirements.

Additionally, you will also be writing a function to evaluate postfix expressions that takes in two arguments; the input expression that is in postfix format, and a stack object. This will use the functionality that you worked on in PA-1 i.e., the infix_to_postfix() as well as the Stack classes you built which will be used to create a stack object. This function should return an integer value as its output for all valid expressions and None for invalid expressions.

Expected Functionality:

This exercise requires you to implement Queue Abstract Data type in Python. You should create Queue using NumPy Arrays and write the code for required operations as asked. The goal of this exercise is to make you familiar with Abstraction and several operations written within.

Starter Code:

You will be provided with queue_using_array.ipynb Python notebook. You should edit the file to implement the operations of a Queue as follows.

__init__(initial_capacity) - The constructor for the Queue class initializing the required structure.
enqueue(data) - This function should be able to add an item in the queue.
dequeue() - This function should remove the front element from the queue and return it.
isEmpty() - This function should return whether the queue is empty or not.
isFull() - This function should return whether the queue is full or not.
peek() - This function should return the topmost element in the queue without removing it.
size() - This function should return the size of the queue.

Rubric:

Your code will be tested with provided test cases.

Location of the code:

The code would be provided at
https://colab.research.google.com/drive/1qYViAa8Y_cLlQt8Oy92RfgN6UP8AjeVp?usp=sharing

What to do when done:

Once you have completed the exercise, you should upload it to the codePost. 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 LE3.py
- Upload the python script file to codePost under the LE-3 assignment along with the PA1.py file
- 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