Homework #3

Due by Sunday 7/26, 11:55 pm

Submission instructions:

1. For this assignment, you should turn in 4 files:
    - 4 '.py' files, one for each question 1-4. Name your files: YourNetID_hw3_q2.py' and 'YourNetID_hw3_q3.py', etc.

      Note: your netID follows an abc123 pattern, not N12345678.

2. You should submit your homework via Gradescope.
    - Name all classes, functions, and methods exactly as they are in the assignment specifications.
    - Make sure there are no print statements in your code. If you have a tester code, please put it in a "main" function and do not call it.

Question 1:

Implement the python function:

```
def createTwoDimensionalArray(row, column, initialValue):
```

This function gets 3 integers as input `row, column, initialValue`.
When called, it should return a list of lists equivalent to a two-dimensional array of having `row` number of rows, `column` number of columns, and each of the cell will have initial value of `initialValue`. `row and column` will be positive integers.

As for example, `createTwoDimensionalArray(3, 4, -1)` will return a list of lists `[[-1, -1, -1, -1], [-1, -1, -1, -1], [-1, -1, -1, -1]]`.

This is equivalent to following the two-dimensional array of having 3 rows, 4 columns, and each cell is initialized
with -1:

```
-1 -1 -1 -1
-1 -1 -1 -1
-1 -1 -1 -1
```

Question 2:

Write a recursive function that creates a deep copy of a nested list. That is, each list in the hierarchy is copied and modification to the original or new list will not affect the other.

```
def deep_copy(lst):
```

Note: Do not use the copy library. You must write the recursive function completely on your own.

Question 3:
Give a Python implementation for the *MaxStack* ADT. The *MaxStack* ADT supports the following operations:

- `MaxStack()`:initializes an empty Max Stack object.
- `maxS.is_empty()`: returns `True` if `maxS` does not contain any elements, or `False` otherwise.
- `len(maxS)`: Returns the number of elements in `maxS`.
- `maxS.push(e)`: adds element `e` to the top of `maxS`.
- `maxS.top()`: returns a reference to the top element of `maxS`, without removing it; an exception is raised if `maxS` is empty.
- `maxS.pop()`: removes and return the top element from `maxS`; an exception is raised if `maxS` is empty.
- `maxS.max()`:returns the element in `maxS` with the largest value, without removing it; an exception is raised if `maxS` is empty.

Note: Assume that the user inserts only integers to this stack (so they could be compared to one another, and a maximum data is well defined). For example, your implementation should follow the behavior below:

```
>>> maxS = MaxStack()

>>> maxS.push(3)

>>> maxS.push(1)

>>> maxS.push(6)

>>> maxS.push(4)

>>> maxS.max()

6

>>> maxS.pop()

4

>>> maxS.pop()

6

>>> maxS.max()

3
```

Implementation Requirements:

1. For the representation of `MaxStack` objects, your data members should be:
   a. A Stack – of type `ArrayStack`
   b. Additional $\theta(1)$ space - for additional data members, if needed

2. Your implementation should support the `max` operation in $\theta(1)$ worst-case time. For all other Stack operation, the running time should remain as it was in the original implementation.

Hint: You may want to store a tuple, as elements of the `ArrayStack`. That is, to attach to every "real" data in this stack some additional information.

Question 4:

Implement the following function
```
def permutations(lst)
```

The function is given a `lst` of integers, and returns a list containing all the different permutations of the elements in `lst`. Each such permutation should be represented as a list.
For example, if `lst = [1,2,3]`, the call could return `[[1, 2, 3], [2, 1, 3], [1, 3, 2], [3, 2, 1], [3, 1, 2], [2, 3, 1]]`

Implementation Requirements:

1. Your implementation should be **non-recursive**.
2. Your implementation is allowed to use a Stack, a Queue, and $\theta(1)$ additional space.

Hint:

2. Use the stack to store the elements yet to be used to generate the permutations and use the queue to store the (partial) collection of permutations generated so far.
3. You might want to make sure that your queue is setup to hold all the partial permutations generated so far.