Using the Stack and Deque ADTs
CMPU 102 Lab 8:

In this lab, you will write a program called TYPESETTER that uses both a stack and a deque. A deque is a linear collection that supports element insertion and removal at both ends. The name deque is short for "double ended queue" and is usually pronounced "deck". You can also do all the Queue operations (enqueue, dequeue, and isEmpty, in particular) on a deque.

Implementing the Postscript™ roll command

Postscript™ is a powerful graphics language that stores data on a stack. One of the PostScript commands is called roll. This command takes two integer arguments, \( n \) and \( k \). The effect of applying \( \text{roll}(n, k) \) is to rotate the top \( n \) elements of a stack by \( k \) positions, where the general direction of the rotation is toward the top of the stack. More specifically, \( \text{roll}(n, k) \) has the effect of removing the top \( n \) elements and enqueuing them on a deque, cycling the top element on the deque to the last position \( k \) times, and then replacing the reordered elements on the stack by removing the reordered \( n \) items from the deque.

Download the zipped starter files from the web page and unzip them into your cs102/labs/lab8/cs102-lab8-files directory. Open the project. The Deque and Stack interfaces are already written as are the StackEmptyException and DequeEmptyException classes. The other classes are started for you.

The figure below shows before and after pictures for three different examples of roll. In these examples, there is one stack, \( S \), and 1 deque, \( D \).

The signature of the roll method is as follows:

```java
void roll( int n, int k ) // The stack is global, no need to pass it as argument
```

that implements the \( \text{roll}(n, k) \) operation on the specified stack. Your implementation should prompt the user for a positive number \( n \) that is not larger than the stack size and a number \( k \) that is nonnegative. Note, however, that \( k \) can be larger than \( n \), in which case, the roll operation continues through more than a complete cycle. This case is illustrated in the rightmost figure above.
You can use any class you want to implement the stack and dequeue (array, linked list, ArrayList). These classes should be separate from the TYPESETTER file but in the same package or in the Java API. You can also choose any input and output classes to read and write data in the program. The starter project uses TextIO.java.

You should add toString() methods to the charStack and charDeck classes (besides the methods they are required to implement for the interfaces) so that the result of a successful operation could look like this:

```
Please enter a string of characters to put on stack.
DIALOG
Please enter the number of items to pop.
3
Please enter the number of times to roll.
? 1
Stack before roll:

| G |
| O |
| L |
| A |
| I |
| D |

---

Deque after 3 pops:

- - -
G O L
- - -

Deque after 1 roll:

- - -
O L G
- - -

Stack after roll:

| O |
| L |
| G |
| A |
| I |
| D |

---
```

Submit the zipped folder "cs102-lab8-yourlastname" containing the solutions and be sure to get checked off for attending lab today.