CS390 Quiz 2

1. For this problem, you will implement a stack of `Integers`, using an array in the background. To help you get started, there is skeleton code provided to you with a class named `MyIntStack`. There are no implemented methods in this class though. Your task is to provide implementations of all of the methods declared there so that your class behaves as a stack. The methods to be implemented are: `isEmpty, push, pop`, and `peek`. The comments provided in the code specify the expected behavior of each of these operations.

   Your code must meet the following requirements:

   a. The `push` method should never cause an exception to be thrown.
   b. The only situation in which `pop` or `peek` should cause an exception to be thrown is if either of these operations is called when the stack is empty. In that case, a `MyStackException` should be thrown. Note that note that `MyStackException` is a class that has been provided for you.
   c. Your stack should reject `null` inputs – it should not be possible to push a `null Integer` onto your stack. If the `push` method is passed a `null` argument, it should do nothing (no exception should be thrown, and the stack should not be modified).

2. You are asked to write a class `MinDoublyLinkedList`. A `MinDoublyLinkedList` is a doubly linked list with header, and one more property - the minimum element is always at the node after header. (We adopt the convention that header is in position -1 and the node after header is in position 0, etc...)

   First, implement an `add` method with the following signature:

   ```
   public void add(String item)
   ```

   The `add` method uses the following strategy: if the element to add is greater than the 0^{th} element, add it right after 0^{th} element; otherwise, the element to add becomes 0^{th} element and the original 0^{th} element becomes 1^{st} element. No changes to any other nodes.
   Example. Suppose your list has these values:

   ```
   ["bill","tom","mike"]
   ```

   After executing `add("anne")`, the list contains these elements (in this order):

   ```
   ["anne","bill","tom","mike"]
   ```

   After executing `add("chris")`, the list contains these elements (in this order):

   ```
   ["anne","chris","bill","tom","mike"]
   ```

   A `toString` method has been provided so you can test your code.

   Then implement the two other methods below.

   ```
   //returns the minimum element in the linkedlist
   public String min()
   //finds the maximum element in the linkedlist and removes it.
   public String removeMax()
   ```

Requirements for this problem:

(1) No data may be placed in the header node.

(2) You may not introduce any new instance variables or instance methods in `MinDoublyLinkedList`.

(3) During execution, each `Node` in your `MinDoublyLinkedList` must have correct values for the `next` and `previous` `Node`s.