Assignment 5:

In addition to the base files, three additional files are attached: *EmptyCollectionException.java*, *LinearNode.java*, and *StackADT.java*. These files will need to be added to your Java project. They provide data structure functionality that you will build over. It is suggested that you test if these files have been properly added to your project by confirming that *Base_A05Q1.java* and *Base_A05Q3.java* compile correctly.

- **Q1:** Complete the implementation of the ArrayStack class presented in Chapter 12. Specifically, complete the implementations of the isEmpty, `size`, and `toString` methods. See *Base_A05Q1.java* for a starting place and a description of these methods. [10 points]

  Sample Output
  ```
  STACK TESTING
  4
  8
  8
  9
  The size of the stack is: 3
  The stack contains:
  9
  7
  3
  ```
- **Q2:** There is a data structure called a drop-out stack that behaves like a stack in every way except that the stack has a maximum capacity *n*. If the stack contains *n* elements, when the n+1 element is pushed, the bottom element is lost. For example, if the stack has a maximum capacity of 3, and contains the values (1, 2, 3), with 3 being the top element, then pushing a 4 would give the stack (2, 3, 4), with 4 being the top element. Implement a drop-out stack using an array. Follow the StackADT interface. See *Base_A05Q2.java* for a starting place. Hint: Can any of the code from Q1 be reused? [20 points]

  Sample Ouput
  ```
  DROP-OUT STACK TESTING

  The size of the stack is: 4

  The stack contains:

  5

  4

  3

  2



  The size of the stack is: 4

  The stack contains:

  8

  7
  ```

```
4

3
```

- **Q3:** Complete the implementation of the LinkedStack class presented in Chapter 13 slides. Specifically, complete the implementations of the `peek`, isEmpty, `size`, and `toString`methods. See *Base_A05Q3.java* for a starting place and a description of these methods. [10 points]

  Sample Ouput

  ```
  STACK TESTING

  The stack contains:

  empty

  4

  8

  8

  9

  The size of the stack is: 3

  The stack contains:

  9

  7

  3
  ```

- **Q4:** The basic implementation of LinkedStack supports adding elements at the front of the list (*push()*). Add to your answer from Q3 by writing a method `public void` pushLast(**T**`element) {...}` that will add at the end (bottom) of the list instead of the beginning. Document your method using javadoc. Notice that this method is not part of the **StackADT**inteface. Thus, you need to warn the programmer in your documentation that the method is non-standard and may result in undesirable behavior. [10 points]
-