1. (0 points) For each of the following data structures, briefly characterize its suitability as the underlying data structure for a queue and for a stack, considering add and remove run-time and memory usage. Be sure to note if there are particular considerations that may make the data structure more or less suitable.

   a. array

   b. ArrayList

   c. Single-linked List

2. (0 points) Explain what would happen to the algorithms and the time complexity of an array implementation of a stack if the top of the stack were at position 0.
3. (0 points) Suppose your boss wants you to implement a list using a queue as the underlying data structure for storing stuff. So the class definition might look like:

```java
public class QueueList<T> implements List<T> {
    private Queue<T> q; // This is the only instance data
}
```

The size method would be:

```java
public int size() { return q.size(); }
```

And the add method would be:

```java
public void add(T data) { q.offer(data); }
```

So, your job (on this exam) is to fill in the code for the following method:

```java
public T get(int index) {
```
4. (0 points) Implement the following methods of a singly-linked, null terminated list. Each method must run in $O(1)$. You can assume head and tail pointers.

```java
public T removeLast() {
}

public T removeFirst() {
}

public T addFirst() {
}

public T addLast() {
}
5. (0 points) Hand trace a queue X through the following operations. Show the state of the queue after each call.

```java
X.enqueue(new Integer(4));
X.enqueue(new Integer(1));
Integer Y = X.dequeue();
X.enqueue(new Integer(8));
X.enqueue(new Integer(2));
X.enqueue(new Integer(5));
X.enqueue(new Integer(3));
Integer Z = X.dequeue();
X.enqueue(new Integer(4));
X.enqueue(new Integer(9));
```
6. (0 points) Implement a stack using a linked list as the backing data structure.

```java
public class Stack<T>{
    private LinkedList<T> store = new LinkedList<T>();

    public void push(T o){
    }

    public T pop(){
    }

    public int size(){
    }
}
```
7. (0 points) Implement a queue using a circular array backing data structure.

```java
public class Queue{
    private Object[] store = new Object[100];
    //add any additional vars here.

    public void enqueue(T o){
    }

    public T dequeue(){
    }

    public int size(){
    }
}
```