1. (3 points) What does the following code do (assume you’re running \texttt{main()})?

```java
public class Node {
    int value;
    Node next;
    public Node (Node nx, int n) {
        value = n; next = nx;
    }
    public static void main (String args[]) {
        Node n1 = null, n2 = null;
        n1 = new Node (n2, 5);
        n2 = new Node (n1, 7);
        System.out.println("Next of n2 is " + n2.next.value);
        System.out.println("Next of n1 is " + n1.next.value);
    }
}
```

The code will print out \texttt{Next of n2 is 5} and then throw a \texttt{NullPointerException}. The reason is that \texttt{n2 == null} when \texttt{n1} is initialized, and dereferencing \texttt{null} throws an exception.

2. (3 points) There is one stack operation that may throw an exception in an array-based implementation but not in a list-based operation. What is it, and under what conditions is the exception thrown?

The operation is \texttt{push()}. An array has a fixed size limit, and a call to \texttt{push()} when the array is full (contains as many elements as the array has space for) will cause an exception.

3. (2 points) List one advantage of arrays over doubly linked lists in implementing the List ADT.

- Arrays have less overhead per item.
- Arrays need not allocate space on each operation.

4. (2 points) List one advantage of doubly linked lists over arrays in implementing the List ADT.

- Lists have no fixed maximum size.
- Lists can delete and insert items more quickly (no need to copy elements).