Assignment 3

COMP 2100
Due: October 4, 2024

No member variables or static member variables should be used in the following recursive methods. They should be entirely self-contained.

1. Consider the following definition of a linked list class:

```
public class LinkedList {
      private static class Node {
            public String value;
            public Node next;
      }

      private Node head;
      …
}
```

Write a static method that, given a reference to the first node in a linked list, prints the values in the list in reverse order, recursively. Your function should make no library calls (except for **`System.out.print()`**) and contain no loops. Use the following method signature.

```
private static void printReverse(Node list)
```

This recursive helper method would be called from the following public method within **`LinkedList`**.

```
public void printReverse() {
      printReverse(head);
}
```

2. Write a recursive static method with no loops or other method calls that returns the sum of the contents of an array of **`double`** values. If the array has length zero (legal in Java), its sum is **`0.0`**. Use the following method signature.

```
public static double sumArray(double[] array, int index)
```

Note: The **`index`** parameter gives the current element under consideration. Thus, the method would initially be called as follows.

```
double answer = sumArray(array, 0);
```

3. Write a recursive static method with no loops or other method calls that returns the largest value in an array of **`double`** values. If the array has length zero (legal in Java), return **`Double.NaN`**. Use the following method signature.

   ```
   public static double largest(double[] array, int index)
   ```

   Note: The **`index`** parameter gives the current element under consideration. Thus, the method would initially be called as follows.

   ```
   double answer = largest(array, 0);
   ```

4. Write a recursive static method, using no loops or methods other than **`charAt()`** and **`length()`**, that, given a **`String`** storing arbitrary text, returns **`true`** if the word is a perfect palindrome, that is, contains the exact same sequence of characters forwards and backwards, and **`false`** otherwise. Use the following method signature.

   ```
   public static boolean isPalindrome(String text, int index)
   ```

   Note: The **`index`** parameter gives the location of the current character under consideration. Thus, the method would initially be called as follows.

   ```
   boolean answer = isPalindrome("racecar", 0); // true
   ```

5. Write a recursive static method using no loops that, given a **`String`** storing a full phrase, returns **`true`** if the phrase is a palindrome and **`false`** otherwise. This method differs from the previous method in that case, spaces, and punctuation (non-letter characters) are ignored. Use the following method signature.

   ```
   public static boolean fullPalindrome(String phrase, int start,
         int end)
   ```

   Note: The **`start`** parameter gives the index of the first character under consideration, and the **`end`** parameter gives the index **after** the last character under consideration. You may use **`Character.toUpperCase()`** and **`Character.isLetter()`** as well as the **`charAt()`** and **`length()`** methods from the **`String`** class but nothing else. The method would initially be called as follows.

   ```
   boolean answer = fullPalindrome("Madam, I'm Adam!", 0, 16); // true
   ```

   You are only required to turn in the text of these methods, not a **`.java`** file that can be compiled. **However, testing them with a Java compiler would be wise.**

   For typesetting code, consider using the **`verbatim`** or **`listings`** packages in LaTeX.