CS1020 SIT-IN LAB 04 B – NUMERIC SEQUENCE

Semester 2 AY2014/2015

Advice

1. Understand the problem thoroughly. Ask the invigilator when unsure.
2. Check out the given input and output files, and use the "diff" command to compare your own output with the expected output.
3. Make sure your code can be compiled ***AT ALL TIMES!***
4. Note that code that is commented out will **not** be looked at and graded.

Important notes:

1. You are to use **recursion**. No mark will be awarded for correctness even if your program passes the test data but does not use recursion in a correct and meaningful manner. Using recursion only for peripheral tasks (eg: reading the input) does not count as meaningful usage.
2. You are not supposed to use any **loop** statements. A solution containing loops will not be awarded any marks.
3. Efficiency of your program will be taken into account in grading.
4. Note that there are **3** test cases for part 1 and **7** test cases for part 2.

Problem Description

Part 1

We want to check if an integer array of numbers is sorted in increasing order or not. You may assume that the array does not contain any duplicate elements.

Input

The first line of input is an integer 1 indicating part 1. The next line contains a positive integer *N* indicating the length of the array. The next line contains *N* integers separated by space to be read into the array.

Two examples are shown below. (Note that comments on the right are for explanatory purpose and they do not appear in the input.)

Input #1

```
1
5
1 3 5 7 9
```

Input #2

```
1
5
1 3 5 9 7
```

The test case is for part 1.
Length of the array.
Values for the array elements.

Output

The program should output either **Maximum element *n*** where *n* is the maximum element of the array, or **Not sorted** if the array is not sorted.

For the above two examples the outputs are:

Output #1

```
Maximum element 9
```

Output #2

```
Not sorted
```

Part 2

For the second part of the problem consider an array of integers where the numbers are **initially increasing** and **then decreasing**. The program needs to find out the maximum number in the array.

As in part 1, you may assume that the array does not contain duplicate elements. Note that for a sequence either the ascending or descending part may be absent.

Input

The first line of input is an integer 2 indicating part 2. The next line contains a positive integer *N* indicating the length of the array. The next line contains *N* integers separated by space to be read into the array.

An example is shown below. (Note that comments on the right are for explanatory purpose and they do not appear in the input.)

```
2
5
10 20 30 25 15
```

The test case is for part 2.
Length of the array.
Values for the array elements.

Output

The program should output **Maximum element *n***, where *n* is the maximum element in the array.

For the above input the output is:

```
Maximum element 30
```

Skeleton program

You are provided with a skeleton program **NumericSequence.java**. Do not create any other file and do not rename this file or the public class of this file. You may change the skeleton code, add methods and classes any way you deem fit.

```
import java.util.Scanner;

public class NumericSequence {

    public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      NumericSequence ns = new NumericSequence();
      ns.find(sc);
    }
}
```

-- **Have fun** !!! ☺ --