LEBANESE AMERICAN UNIVERSITY
School of Arts and Science
Department of Computer Science and Mathematics

CSC 310: Algorithms and Data Structures

Lab VI

Important Note: All problems should be read from the same input called "sort.in". The rest of the problems should be preceded by its number "problem1.in".

Problem 1

Given an array of integers, write a program that sorts the array using Radix Sort.

The algorithm works by obtaining the number with the most significant digits, then proceeds to sort the numbers starting from the ones then tens then hundreds and so on. It places the numbers in a list that corresponds to their digit, and empties them back into the array.

Input

The first line of input is an integer T representing the number of test cases.
Each test case is made up of two lines. The first line has an integer N representing the number of elements in the array. The second line contains N integers representing the values in the array.

Output

For each test case, output the sorted array.

Sample Input

```
3
7
25 13 10 30 15 27 37
4
6 7 8 9
6
10 7 15 13 4 6
```

Sample Output

```
10 13 15 25 27 30 37
6 7 8 9
4 6 7 10 13 15
```

Problem 2

Implement a Max-heap data structure (the parent is always larger than its children) using an array. More specifically, you are asked to implement the insert and heapify methods.

Input

Each test case is made up of two lines. The first line has an integer N representing the number of elements in the array. The second line contains N integers representing the values in the array.

Output

For each test case, the heap array.

Sample Input

```
2
5
3 2 1 5 4
4
6 7 8 9
6
10 7 15 13 4 6
```

Sample Output

```
5 4 3 2 1
9 8 6 7
15 13 7 10 4 6
```

Problem 3

Given an array of integers, write a program that sorts the array using Heap Sort.

Using the above Max-heap structure you implemented, you can sort the array of length *n* as follows:

1. Swap the first element(Largest element) in the array with the n^{th} position .
2. Heapify the array on the $n - 1$ elements.
3. Decrement *n* until *n* becomes 0.

Input

Each test case is made up of two lines. The first line has an integer N representing the number of elements in the array. The second line contains N integers representing the values in the array.

Output

For each test case, output the sorted array.

Sample Input

```
2
7
25 13 10 30 15 27 37
4
6 7 8 9
```

Sample Output

```
10 13 15 25 27 30 37
6 7 8 9
```

<u>Problem 4</u>

Write a method that counts the number of nodes in a tree.

<u>Input</u>

Each test case is made up of one line. The integer *N* representing the number of elements in the tree then *N* elements follows.

<u>Output</u>

For each test case, output the number of nodes.

Sample Input

```
3
7 25 13 10 30 15 27 37
4 6 7 8 9
6 10 7 15 13 4 6
```

Sample Output

```
7
4
6
```

<u>Problem 5</u>

Write a method that computes the height of a tree knowing the height at the root is 0.

<u>Input</u>

Each test case is made up of one line. The integer *N* representing the number of elements in the tree then *N* elements follows.

<u>Output</u>

For each test case, output the height of a tree.

Sample Input:	**Sample Output:**
3	
7 25 13 10 30 15 27 37	2
4 6 7 8 9	3
6 10 7 15 13 4 6	3

Problem 6

Write a method that to check whether a tree is a complete. A complete tree has neither 2 children or none. If the tree is complete, print to screen "**I am a complete tree!**" otherwise print "**I am not a complete tree!**"

Input

Each test case is made up of one line. The integer *N* representing the number of elements in the tree then *N* elements follows.

Output

For each test case, if the tree is complete or not.

Sample Input	**Sample Output**
3	
7 25 13 10 30 15 27 37	I am a complete tree!
4 6 7 8 9	I am not a complete tree!
6 10 7 15 13 4 6	I am not a complete tree!

Problem 7

Write a method to check if a tree is AVL by checking if it has balanced heights [-1, 0, 1]. Print "**Is AVL**" if the tree is an AVL tree otherwise print "**Is Not AVL"**.

Input

Each test case is made up of one line. The integer *N* representing the number of elements in the tree then *N* elements follows.

Output

For each test case, if the tree is AVL or not.

Sample Input

```
3
7 25 13 10 30 15 27 37
4 6 7 8 9
6 10 7 15 13 4 6
```

Sample Output

```
Is AVL
Is Not AVL
Is Not AVL
```

Problem 8

Write a recursive method that finds the length of the LCS(Longest Common Substring). The Substring must be continuous. It is the longest group of elements between two groups A and B that are common between the two groups and in the same order in each group. For example:

A: **32**34
B: 1224533**32**4

A: thisisa**test**
B: **test**ing123testing

Input

Each test case is made up of 2 Strings.

Output

For each test case, output the Longest Common Substring.

Sample Input:

```
3
karim
yassine
karim
omar
thisisatest
testing123testing
```

Sample Output:

```
1
1
4
```