Data Structures and Algorithms

Computer Science Degree - Group I

First Semester. January 25, 2018.

Name: _____ **Group:** _____

Laboratory: _____ **PC:** _____ **DOMjudge User:** _____

We say a sorted array of integers has a *staircase* shape if its elements occur e_i times, forming *steps*. We define the *width* of a step as the number of times ($e_i > 0$) that the element of that step occurs in the array. The next array has a staircase shape, with 6 steps of widths 4, 2, 4, 1, 2 and 1:

1	1	1	1	2	2	3	3	3	3	4	7	7	8

but the following one does not have a staircase shape, since its elements are not sorted.

2	2	1	1	1	3	3

We say a staircase has steps of *growing width* if the number of elements of each step is greater than or equal to the number of elements of the previous step.

1	1	2	2	3	3	3	3	4	4	4	4	4	4

1. (4 points) Code an iterative algorithm that, given an array v of integers of length $0 < n \leq 1000$ that represents a valid staircase, returns whether it is a staircase with steps of growing width. In addition to coding the algorithm, you have to specify its precondition, postcondition, invariant and termination function of its loops and calculate its complexity.

Input	**Output**
n v	
3 1 2 2	SI
3 1 2 3	SI
3 1 1 2	NO
3 1 1 1	SI
6 2 2 4 4 6 6	SI
6 2 3 3 4 4 5	NO
1 3	SI
2 3 3	SI
0	

2. (3 points) Code a D&C algorithm that, given an array v of length $0 < n \leq 1000$ with a staircase shape, returns the width of the first step, the last one and the widest one.

Note: the steps don't need to have growing width.

Input	**Output**
n v	
3 1 2 3	1 1 1
4 1 2 2 3	1 1 2
5 1 1 2 3 3	2 2 2
4 1 1 1 1	4 4 4
6 1 1 1 2 2 2	3 3 3
6 1 1 1 1 2 2	4 2 4
6 1 3 3 3 4 4	1 2 3
0	

3. (3 points) Code an algorithm that generates all the possible staircases of length n ($0 < n \leq 30$) which are valid for seniors. A staircase is valid for seniors when:

- The first step has a height of 1.
- The height of all the steps is 1 (i.e. the height of the second step is 2, the height of the third one is 3, and so on).
- The steps have growing width.

Input	**Output**
n	
2	1 1
	1 2
3	1 1 1
	1 2 2
	1 2 3
4	1 1 1 1
	1 1 2 2
	1 2 2 2
	1 2 3 3
	1 2 3 4
0	