Data Structures and Algorithms

Computer Science Degree - Group I

First Semester. February 9, 2017.

Name: _____ **Group:** _____

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

The following definitions will be used in the exercises:

- We say an array is *sorted in ascending order by a hairbreadth* when, in addition to being sorted in ascending order, the difference between an element and the next is ≤ 1.

  For example, the following arrays of size 4 meet the definition:

1	2	3	4

1	2	2	3

1	1	1	1

  Please note that, since it does not need to be sorted in *strictly ascending order*, the last array filled with `1`'s meets the definition.

  On the contrary, the array

1	2	1	2

  is not in *ascending order by a hairbreadth*, since it is not in ascending order.

- Boring arrays are the ones which contain elements that occur many times. We say an array is *d-entertaining* when there is not any element that occurs more than d times.

  Examples of *1-entertaining* arrays are

1	2	3	4

4	3	7	0

  since no element occurs more than once (please note that, for example, these arrays are also *10-entertaining*, since no element occurs more than 10 times).

  Other examples of arrays and their smallest entertaining degree are:

6	7	6	6	*3-entertaining*

7	3	7	0	*2-entertaining*

7	3	7	3	*2-entertaining*

9	9	9	9	*4-entertaining*

We can combine both definitions. For example, the array

1	1	2	2	2	3	4	5

is in *ascending order by a hairbreadth* and *3-entertaining*.

1. (4 points) Specify, design and code a function that receives an array of integers of length $0 \leq n \leq 1000$ and a parameter $d > 0$ and returns whether the array is in *ascending order by a hairbreadth* and *d-entertaining* or not. Write the invariant and the termination function that allow to prove the correctness of the coded function. Justify the cost of your algorithm.

The first line of the input contains the number of test cases. Each test case contains the values of d, n and the elements of the array. The program will write `SI` if the array is in *ascending order by a hairbreadth* and *d-entertaining* and `NO` otherwise.

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

2. (3 points) Design and code a recursive algorithm that receives and array sorted in ascending order (i.e. not strictly) of length $0 \leq n \leq 1000$ and returns whether it is in *ascending order by a hairbreadth* or not. Justify the cost of your algorithm.

Higher marks will be awarded to those solutions that use a *divide and conquer* strategy and than don't traverse the array completely if it is not necessary.

The first line of the input contains the number of test cases. Each test case contains the values of n and the elements of the array. The program will write `SI` if the array is in *ascending order by a hairbreadth* and `NO` otherwise.

Input					**Output**
n	v				
4	1	1	1	1	SI
4	1	1	1	2	SI
4	1	1	1	9	NO
4	1	1	2	2	SI
4	1	2	3	4	SI
4	1	2	3	3	SI
4	1	1	3	3	NO
4	1	2	4	5	NO

3. (3 points) Code a function that generates arrays sorted in *ascending order by a hairbreadth* and *d-entertaining*. The function will receive the size of the array $0 \leq n \leq 1000$, the value of $d > 0$ and the first element of the array e, and it will generate the output in lexicographical order. You can use auxiliary functions.

The first line of the input contains the number of test cases. Each test case contains the values of n, d and e. The program will write each resulting array in a separate line, and the elements will be separated by a blank space.

Input			**Output**
n	d	e	
3	3	1	1 1 1
			1 1 2
			1 2 2
			1 2 3
3	2	1	1 1 2
			1 2 2
			1 2 3
3	2	2	2 2 3
			2 3 3
			2 3 4

.

```
// Writes all the arrays in ascending order by a hairbreadth of size n
// which are d-entertaining and where the first element is e
void writeAscendingByAHairbreadthEntertaining(int n, int d, int e);
```