1) (10 pts) DSN (Recursive Coding)

The longest increasing subsequence problem is as follows: given a sequence of integers, find the largest subset of those integers such that within the sequence, those integers are in increasing order. For example, for the sequence **2**, 9, 4, **3**, 7, **5**, **6**, **8**, has some increasing subsequences of length 5 (one of these is highlighted) but none of length 6, so the length of the longest increasing subsequence of this sequence is 5.

In order to solve this problem recursively, we have to reformulate the problem a little bit. Namely, our input will be:

1. An array, ***values***, storing the original sequence
2. An integer, ***k***, representing that we want to only consider the values in the array upto index k, including it.
3. An integer, ***max***, representing the maximum value allowed in the increasing sequence.

Our recursive function will return the length of the longest increasing subsequence of ***values***[0..***k***] such that no value in the increasing subsequence exceeds max.

Complete the implementation of this ***recursive*** function below:

```
int lis(int* values, int k, int max) {

    if (k == 0) return values[0] <= max;

    int notake = lis(values, k-1, max);

    int take = 0;
    if (values[k] <= max)
        take = 1 + lis(values, k-1, values[k]-1);

    if (take > notake)
        return take;
    return notake;

}
```

Grading:

3 pts base case (there are other options here…we can use k = -1 for example, or nest a second if in the k == 0 one.

2 pts no take case - two points for this recursive call, don't take off if values is omitted…1 pt off per other two parameters

4 pts take case - 1 pt for checking if we can take the last value, 1 pt for the recursive call, 1 pt for a new max, 1 pt for adding 1

1 pt - return maximum of the two cases