Assignment 3

Exercise 1

Design and implement a program in C, that given a sequence $s = [e_0, e_1, \ldots, e_{n-1}]$ of numbers, and a number x, it checks whether there exists a subset of the elements in s whose sum equals x. This problem must be solved *recursively*, i.e., the function that checks whether there is a subset whose sum leads to x must be a recursive function. As an example, if the sequence is $s = [-73, 30, 44, 23, -40, -92, 87]$ and the value is 20, the function must return *false* (zero), since no subset of the sequence has sum 20. On the other hand, if for the same sequence, the value we consider is 27, the function must return *true* (one), since $44 + (-40) + 23 = 27$.

Your task is to extend the partial implementation in file `sum_exists.c`, with a recursive implementation of function `sum_exists(seq, lower, upper, n)`. Follow the comments that accompany the function to have more precise details of what this function must implement.

The main function in the file receives a seed and the value x from the command line, it generates a random sequence using the seed, and then checks whether the generated sequence contains elements whose sum is x, using the function you have to implement. You can use the overall implementation to test your solution. You may fully disregard efficiency for this exercise, and concentrate on functional correctness, as well as code clarity.

Exercise 2

Design and implement a program in C, that given a sequence $s = [e_0, e_1, \ldots, e_{n-1}]$ of numbers, computes and prints out the longest subsequence of s that is strictly increasingly sorted. For instance, if the sequence is $s = [-73, -73, 30, 44, 23, -40, -92, 87]$, then the longest strictly increasing subsequence is $[-73, 30, 44]$. This problem can be solved iteratively or recursively.

Your task is to extend the partial implementation in file `strictly_increasing_subsequence.c`, with an implementation of function `strictly_increasing_subsequence(seq, lower, upper, result)`. Follow the comments that accompany the function to have more precise details of what this function must implement.

The main function in the file receives a seed from the command line, it generates a random sequence using the seed, and then computes and prints the longest strictly increasing subsequence of the generated sequence, using the function you have to implement. You can use the overall implementation to test your solution. You may fully disregard efficiency for this exercise, and concentrate on functional correctness, as well as code clarity.