Problem 1

Given is a sequence of numbers $a_1, a_2, a_3, \ldots, a_n$. We say that $a_{j_1}, a_{j_2}, \ldots, a_{j_k}$ is a subsequence of this sequence iff $1 \leq j_1 < j_2 < \ldots < j_k \leq n$. The subsequence is increasing iff $a_{j_1} < a_{j_2} < \ldots < a_{j_k}$. In the longest increasing subsequence problem we search for an increasing subsequence of $a_1, a_2, a_3, \ldots, a_n$ that is the longest possible (i.e, $k$ is as large as possible).

For example, for sequence $8, 2, 5, 7, 3, 4, 9, 6, 10$, the longest increasing subsequences are $2, 3, 4, 6, 10$ and $2, 5, 7, 9, 10$ – either of them is a valid longest increasing subsequence.

Mr. Brilliant came up with the following greedy approaches to the problem:

- Find the smallest number in the sequence – if there are more than one, select the one with the smallest index. Suppose the number is $a_\ell$. Output the number. Repeat the previous steps for the input sequence $a_{\ell+1}, a_{\ell+2}, \ldots, a_n$, until the input sequence is empty.

- Try the following with $\ell = 1, 2, \ldots, n$: Let $i = \ell$. Include $a_i$ in the subsequence. Find the smallest $j$ such that $a_i < a_j$. If there is no such $j$, stop. Else, let $i = j$ and repeat the steps described in the last two sentences. Once you are done trying all possible $\ell$’s, out of the $n$ obtained subsequences output one with the longest length.

For every greedy approach:

(a) Give a corresponding detailed and properly structured pseudo code. In particular, the pseudo code should be like code, without worrying about syntactical issues like semi-colons. Do not use any go-to statements, breaks, continues, and exceptions.

(b) Determine whether the approach works. Does it correctly identify a longest subsequence for every input? If not, provide a counterexample. In particular, provide 1. the input sequence, 2. an optimal solution, and 3. the solution produced by the greedy approach.

Problem 2

This problem is also about the longest increasing subsequence problem (see Problem 1). You will implement a recursive approach and a dynamic-programming-based approach and compare their running times. In both cases we are interested only in finding the length of the sequence, not the sequence itself.

- Implement the following recursive approach. Implement the function $\text{incrSubseqRecursive}(j, A)$, that computes the maximum length of an increasing subsequence of the sequence $a_1, a_2, \ldots, a_n$ (stored in the array $A$) that ends with the element $a_j$. This function tries to find a previous element $a_i$ such that $i < j$ and $a_i < a_j$, and then it recursively searches for the maximum length of an increasing subsequence of $a_1, a_2, \ldots, a_i$. It tries up to $j - 1$ different $i$’s and it chooses the one that gives rise to the maximum length. By concatenating $a_j$ to the subsequence of $a_1, a_2, \ldots, a_i$ ending with $a_i$, we get a longest increasing subsequence of $a_1, a_2, \ldots, a_j$. 


• Implement the dynamic programming approach on the last slide of the interval scheduling / longest increasing subsequence slides.

• Generate about 10 inputs for different values of $n$ (how large can $n$ be?) and report your observations on the running times of the two respective algorithms.

Problem 3

Given is a sequence of integers $a_1, a_2, \ldots, a_n$ and a positive integer $k$. Give an $O(n^2)$ algorithm that finds a longest possible subsequence of $a_1, a_2, \ldots, a_n$ such that no two consecutive elements of the subsequence have the same remainder when divided by $k$.

For example, for sequence 1, 5, 6, 2, 4, 7 and $k = 2$, we need to find a longest subsequence that alternates odd and even elements. One of such longest subsequences is 5, 2, 7.

As another example, consider sequence 1, 5, 6, 2, 4, 7 and $k = 3$. Then, a longest subsequence that follows the remainder after dividing by $k$ requirement is 1, 5, 6, 2, 7.

Problem 4

Given is a sequence of integers $a_1, a_2, \ldots, a_n$ and a positive integer $k$. Give an $O(n^2)$ algorithm that finds the number of all subsequences of $a_1, a_2, \ldots, a_n$ such that no two consecutive elements of the subsequence have the same remainder when divided by $k$.

For example, for sequence 1, 5, 6, 2, 4, 7 and $k = 2$, all such subsequences are

• of length one: 1, and 5, and 6, and 2, and 4, and 7 (a total of 6 subsequences of length one),

• of length two: 1, 6, and 1, 2, and 1, 4, and 5, 6, and 5, 2, and 5, 4, and 6, 7, and 2, 7, and 4, 7 (a total of 9 subsequences of length two), and

• of length three: 1, 6, 7, and 1, 2, 7, and 1, 4, 7, and 5, 6, 7, and 5, 2, 7, and 5, 4, 7 (a total of 6 subsequences of length three).

Hence, the total number of such subsequences is 21.