Problem 1 (10pt)

Describe the activity selection problem.
Problem 2 (30pt)

Consider a modified version of the Activity Selection problem, where we have a set of activities \( A = \{a_1, \ldots, a_n\} \), and each activity \( a_i \) takes place in the open interval \([s_i, f_i)\). Differently from the standard formulation, here we want to maximize the class utilization, i.e. the amount of time the class is utilized.

Provide the pseudo code of a greedy algorithm for this problem (20pt), highlighting what is the greedy criteria adopted at each iteration. Provide a counter example (10pt), showing the output of the proposed algorithm and the optimal solution.
Problem 3 (25pt)

A stubborn professor wants to prove that the greedy approach to the activity selection problem based on earliest starting time always provide an optimal solution. This is the second step of his proof, discuss why the proof is not correct. You should identify the wrong logical step, discuss why it is not correct and provide a counter example in which the logical step is wrong.

**NOTE:** You should not just provide a counter example in which earliest starting time does not provide an optimal solution.

Let $S_h$ be the solution at iteration $h$, with $h = 0, \ldots, m$ and $m$ is the number of iterations of the while loop.

Base case: $h = 0$, $S_0 = \emptyset \subseteq S^*$.

Inductive Hypothesis: Until iteration $h$, there exist an optimal solution $S^*$ s.t. $S_h \subseteq S^*$.

Inductive step: Let $a_k$ be the element selected by the algorithm at iteration $k$, therefore $S_{h+1} = S_h \cup \{a_k\}$.

If $a_k$ belongs to $S^*$, we are done. Otherwise let's select $a_j \in S^* \setminus S_h$ as the activity with the earliest starting time. We now build $S^\# = (S^* \setminus \{a_j\}) \cup \{a_k\}$. Since $a_j$ does not create overlap in $S^*$ then it does not create overlap also in $S_h$. Therefore, by removing $a_j$ from $S^*$ and adding $a_k$ we do not create an overlap.

Since $|S^\#| = |S^*|$, $S^\#$ is optimal, and $S_{h+1} \subseteq S^\#$, we found the optimal solutions we were looking for.
Problem 4 (25pt)

Provide an efficient implementation (detailed pseudo-code) for the greedy approach based on selecting the activity with minimum number of overlapping activities and discuss its complexity.

*Note: the number of overlapping activities for each activity should be explicitly calculated.*
Problem 5 (10pt)

Consider the following instance of the Knapsack problem, find the output of the greedy approach based on maximum weight and discuss the result in comparison with the optimal solution. The Knapsack has capacity $W = 20$.

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$a_2$</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>$a_3$</td>
<td>100</td>
<td>3</td>
</tr>
<tr>
<td>$a_4$</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>