Problem 1. If we run the activity-selection algorithm taught in the class on the following input:
\[ S = \{[1, 10], [2, 22], [3, 23], [20, 30], [25, 45], [40, 50], [47, 62], [48, 63], [60, 70]\} \]
what is the set of intervals returned?

Problem 2. The following is an alternative greedy algorithm for solving the activity selection problem. Initialize an empty \( T \), and then repeat the following steps until \( S \) is empty:

- (Step 1) Add to \( T \) the interval \( I = [s, f] \) in \( S \) that has the largest \( s \)-value.
- (Step 2) Remove from \( S \) (i) the interval \( I \), and (ii) all the intervals that overlap with \( I \).

Finally, return \( T \) as the answer.

Prove: the above algorithm returns an optimal solution. (Hint: think how to cleverly relate the above algorithm to the algorithm taught in the class. Then you would realize there is a two-word proof for this problem: “by XXXXXXXX”, where XXXXXXXX is an 8-letter English word.)

Problem 3 (0-1 Knapsack). Suppose that there are \( n \) gold bricks, where the \( i \)-th piece weighs \( p_i \) bounds and is worth \( d_i \) dollars. Given a positive integer \( W \), our goal is to find a set \( S \) of gold bricks such that

- the total weight of the bricks in \( S \) is at most \( W \), and
- the total value of the bricks in \( S \) is maximized (among all the sets \( S \) satisfying the first condition).

Assuming \( d_1 \geq d_2 \geq ... \geq d_n \), let us consider the following greedy algorithm:

1. \( S = \emptyset \)
2. for \( i = 1 \) to \( n \)
3. if \( p_i \leq W \) then
4. add \( p_i \) to \( S \); \( W \leftarrow W - p_i \)

Prove: the above algorithm does not guarantee finding the desired set \( S \).