1. This question is concerned with the max subarray problem.
   Apply the dynamic programming algorithm to the array \([-3, 10, -8, 9]\).

2. Show that the naive algorithm for the max-subarray problem is in \(O(n^2)\),
   and that the dynamic programming algorithm for the max-subarray problem is in \(O(n)\).

3. This question is concerned with rod cutting.
   Apply the dynamic programming algorithm to the price table from the slides with \(n = 3\).

4. The recurrence equation for the recursive algorithm for rod cutting is given by
   \(T(n) = 1 + \sum_{j=0}^{n-1} T(j)\) together with \(T(0) = 1\). Use induction to show
   that \(T(n) = 2^n\).

5. This question is concerned with rod cutting.
   We adapt the problem: now there is a fixed cost of \(c\) per cut. The benefit
   is the sum of the values of the rods, minus the costs for cutting. Apply
   the dynamic programming algorithm to deal with this situation.

6. This question is concerned with rod cutting.
   Give an alternative algorithm for rod cutting that works top-down but
   reuses sub-results.

7. This question is concerned with knapsack01.
   Apply the algorithm for knapsack01 to the following set \(S\).

   \[
   \begin{array}{c|cc}
   s & b & w \\
   \hline
   s_1 & 2 & 1 \\
   s_2 & 5 & 2 \\
   s_3 & 4 & 3 \\
   \end{array}
   \]

   with maximal total weight \(W = 5\).

8. This question is concerned with knapsack10.
   Apply the algorithm for knapsack01 with

   \[
   S = \{ (12, 4), (10, 6), (8, 5), (11, 7), (14, 3), (7, 1), (9, 6) \}
   \]

   and \(W = 18\).
9. This question is concerned with knapsack01.
Adapt the algorithm for knapsack01 such that it does not only give the best possible benefit, but also the subset of items yielding the best possible benefit.
Apply this adapted algorithm to the following set $S$ of items $s_i$ with benefit $b_i$ and weight $w_i$:

$$
\begin{array}{|c|c|c|}
\hline
s_i & b_i & w_i \\
\hline
s_1 & 2 & 1 \\
s_2 & 6 & 4 \\
s_3 & 10 & 5 \\
s_4 & 3 & 2 \\
\hline
\end{array}
$$
with maximum total weight $W = 8$.

10. This exercise is concerned with knapsack01.
Given a set $S$ for the knapsack01 problem with the following property: the order of the items if sorted according to increasing weight ($w$) is the same as the order of the items if sorted according to decreasing benefit ($b$). Example:

$$
\begin{array}{|c|c|c|}
\hline
s_i & b_i & w_i \\
\hline
s_1 & 3 & 1 \\
s_2 & 2 & 2 \\
s_3 & 1 & 3 \\
\hline
\end{array}
$$
Give an efficient algorithm for this special case of knapsack01.