Musterlösung zur 2. Aufgabe:
Dynamisches Progammieren.

Definition der Größen, die in der dynamischen Programmierung auftreten: $l_i^+ :=$ Länge l der längsten Zickzack-Teilfolge von $a_1, \ldots, a_i$, die mit a_i aufhört, und wo der letzte Schritt *aufsteigend* ist (falls $l > 1$ ist).

$l_i^- :=$ Länge l der längsten Zickzack-Teilfolge von $a_1, \ldots, a_i$, die mit a_i aufhört, und wo der letzte Schritt *absteigend* ist (falls $l > 1$ ist).

Rekursionsgleichungen:

$$l_i^+ := 1 + \max\,\{0\} \cup \{\, l_j^- \mid 1 \le j < i,\ a_j < a_i \,\} \quad (1 \le i \le n)$$

$$l_i^- := 1 + \max\,\{0\} \cup \{\, l_j^+ \mid 1 \le j < i,\ a_j > a_i \,\} \quad (1 \le i \le n)$$

(Begründung!)

Lösung des ursprünglichen Problems: Länge L der längsten Teilfolge

$$L = \max\{l_1^+, l_2^+, \ldots, l_n^+, l_1^-, l_2^-, \ldots, l_n^-\}$$

Algorithmus: Lösen der Rekursionsgleichungen für l_i^+ und l_i^-, für $i = 1, \ldots, n$, und Berechnung von L gemäß obiger Formel. Anschließend Rückverfolgung der optimalen Lösung zur Bestimmung der längsten Folge.

Speicher $O(n)$, Laufzeit $O(n^2)$.

Für diese Lösung hätte es die volle Punktzahl gegeben. (Den Algorithmus müsste man noch ein bisschen expliziter ausformulieren.)

Vereinfachungen und Verbesserungen: 1. Beobachtung: Man kann ohne Beschränkung der Allgemeinheit annehmen, dass das letzte Element a_n zur Teilfolge gehört:

$$L = \max\{l_n^+, l_n^-\}$$

(Dies ändert nichts an der asymptotischen Laufzeit.)

2. Der folgende Algorithmus liefert eine optimale Lösung, wenn die Eingabefolge keine zwei aufeinanderfolgenden Elemente enthält, die gleich sind:

> Wähle a_1 und a_n aus, und alle Elemente a_i $(1 < i < n)$, für die $a_{i-1} < a_i > a_{i+1}$ oder $a_{i-1} > a_i < a_{i+1}$ ist.

Die ausgewählten Elemente bilden dann eine längste Zickzack-Teilfolge T^*. Dies führt auf einen *greedy*-artigen Algorithmus mit linearer Laufzeit. Man kann den Algorithmus auch so formulieren:

> Streiche alle Elemente a_i $(1 < i < n)$ mit $a_{i-1} < a_i < a_{i+1}$ oder $a_{i-1} > a_i > a_{i+1}$.

Beweisskizze für die Optimalität von T^*: Betrachte eine längste auf- oder absteigende aufeinanderfolgende Teilfolge, z. B. $a_i < a_{i+1} < \cdots < a_j$. (Die Elemente a_i und a_j sind dann zwei aufeinanderfolgende Elemente von T^*.) Jede Zickzack-Teilfolge T kann höchstens zwei Elemente aus den Elementen $a_i, a_{i+1}, \ldots, a_j$ enthalten. Wenn T zwei dieser Elemente enthält, dann kann man diese beiden Elemente durch a_i und a_j ersetzen, und erhält eine gleich lange Zickzack-Teilfolge. (Warum?) Wenn T eines dieser Elemente enthält, dann kann man dieses Element durch a_i oder a_j ersetzen.

Indem man dieses Argument auf alle monotonen aufeinanderfolgende Teilfolgen anwendet, kann man somit jede beliebige Zickzack-Teilfolge T in eine gleich lange Zickzack-Teilfolge $T' \subseteq T^*$ umwandeln, die nur ausgewählte Elemente enthält.

Man kann den Algorithmus auch leicht auf den Fall ausdehnen, wo gleiche Elemente $a_i = a_{i+1}$ in der Eingabe vorkommen. (Übung)