1.

Algorithmus Auxilary
Input int i, was entweder 0, 1 oder 2 ist
Output int, was gleich int i, falls i < 2, sonst -1 ist

```
1:  if i == 0 OR i == 1
2:     return i
3:  else
4:     return -1
5:  end if
```

Algorithmus The Part Nobody Cares About
Input int length, was die Länge des jetztigen Arrays ausdrückt
Output int n0, int n1, int n2, was die Indezies sind um das Array in 3 zu teilen

```
1:  remainder = length % 3
2:  n0 = 1
3:  n1 = int(((length - Auxilary(remainder)) / 3)) + 1
4:  n2 = int(2 * int((length - Auxilary(remainder)) / 3)) + 1
5:  return n0, n1, n2
```

Um die beiden Hilfsfunktionen oben zu erläutern, hier einige Beispiele:

$$\begin{aligned}
[100,100,100,100,100,100,100,100,100,100] &\Rightarrow \texttt{length} = 9; \quad \texttt{remainder} = 0 \\
&\Rightarrow \texttt{n0} = 1; \quad \texttt{n1} = 4; \quad \texttt{n2} = 7 \\
[100,100] &\Rightarrow \texttt{length} = 2; \quad \texttt{remainder} = 2 \\
&\Rightarrow \texttt{n0} = 1; \quad \texttt{n1} = 2; \quad \texttt{n2} = 3 \\
[100,100,100,100,100] &\Rightarrow \texttt{length} = 5; \quad \texttt{remainder} = 2 \\
&\Rightarrow \texttt{n0} = 1; \quad \texttt{n1} = 3; \quad \texttt{n2} = 5
\end{aligned}$$

Algorithmus Elementary
Input

- Array A der Länge n bevölkert mit 100 bis auf einer Zelle, die ist 101
- index i mit Standardargument 1

Output int, was als Index die Zelle, in der sich 101 befindet, angibt

```
1:  if n <= 1
2:     return i
3:  end if
4:  n0, n1, n2 = The Part Nobody Cares About(n)
5:  if sum(A[:n1]) > sum(A[n1:n2])
6:     return Elementary(A[:n1], i + n0)
7:  else if sum(A[:n1]) < sum(A[n1:n2])
8:     return Elementary(A[n1:n2], i + n1)
9:  else if sum(A[:n1]) == sum(A[n1:n2])
10:     return Elementary(A[n2:], i + n2)
11:  end if
```

Begründung: Da wir bei jeder Rekursion (also bei jeder Wiegeoperation) das Array A in drei (bis auf Rundungsfehler) gleich lange Teilarrays teilen, können wir dieses Problem als eine ternäre Baumdiagramm auffassen. Die Frage dann ist wie hoch dieses Baumdiagramm ist. Da die Blätter aus einzellige Arrays bestehen, haben wir mit h als die Höhe des Baumes

$$3^{\mathrm{h}} > n \iff h > \frac{\log(n)}{\log(3)}.$$

Das bedeutet, dass wir höchstens $\log(n)$ Wiegeoperation benötigen, also kommen wir mit $\mathcal{O}(\log n)$ Operationen aus.

2.

Nummeriere die 10 Kisten von 0 bis 9. Entnehme von jeder Kiste x Äpfel, wobei x die Nummer auf der Kiste ist. Also von der 0. Kiste werden 0 Äpfel entnommen, von der 1. Kiste 1 und so weiter. Insgesammt sind es dann $0+1+2+3+4+5+6+7+8+9 = 45$ Äpfel. Nötige den Spiegel diese 45 Äpfel zu wiegen und sei dieser Wert y. Berechne $y - 4500$. Dieser Wert gibt die Kiste mit den vergifteten Äpfel an, also ist zum Beispiel der Wert 0, so befinden sich die vergifteten Äpfel in der 0. Kiste.

Im schlimmsten Fall sind die vergifteten Äpfel in der 9. Kiste aus der 9 Äpfel entnommen wurden. Daher, falls die sieben Zwerge aus sieben Zwergen bestehen, sollten die übrigen sieben oder mehr Äpfel ausreichen um diese zu assassinieren.