Aufgabe 3.1 (4 Punkte)
Für einen Rechenprozess möchten wir die Anzahl an Cache-Fehlern minimieren. Führe den Algorithmus aus der Vorlesung auf der Instanz $(6, K, \vec{a})$ durch. Der Cache K hat Größe 3. Zu Beginn gilt $K = \{1, 4, 5\}$. Die Sequenz $\vec{a}$ ist gegeben durch

i	1	2	3	4	5	6	7	8	9	10
a_i	2	2	6	1	3	4	1	5	2	6

.

Gib die Datenstrukturen D_1 bis D_{11} wie in Beispiel 3.5 im Skript in Form einer Tabelle an. Unterstreiche jeweils das Informationspaket mit dem maximalen Schlüsselwert. Gib zudem, wie im Beispiel, an, ob ein Cache-Fehler aufgetreten ist.

Aufgabe 3.2 (4 Punkte)
Sei $k \in \mathbb{N}$. Wir betrachten eine Variante *Look-Ahead-k* des Algorithmus zur Minimierung der Cache-Fehler. Das in der Vorlesung vorgestellte Verfahren kann auf die Strategie S_{FF} zurückgreifen, da die Sequenz $\vec{a}$ komplett eingesehen werden kann. Bei der Variante Look-Ahead-k ist die Vorausschau begrenzt. In Runde i kann das Wissen über die nächsten k Einträge von $\vec{a}$ genutzt werden, also $a_{i+1}, \ldots, a_{i+k}$. Liegt ein Cache-Fehler vor, entscheidet er sich wie folgt für das Element, welches aus dem Cache entfernt werden soll: Sind alle Elemente des Cache in den nächsten k Einträgen von $\vec{a}$ zu finden, entferne das Element dessen nächstes Vorkommen so spät wie möglich ist. Gibt es Elemente des Cache, die nicht in der Vorausschau vorkommen, entscheide dich unter diesen für dasjenige dessen Schlüsselelement am kleinsten ist.
Betrachte nun die Eingabeinstanz $(5, K, \vec{a})$, wobei K ein Cache der Größe 3 ist. Zu Beginn gelte $K = \{a_1, a_2, a_3\}$. Zeige durch eine geeignete Wahl $\vec{a}$, dass die Look-Ahead-3-Variante beliebig schlechter abschneiden kann als der Standardalgorithmus mit FF-Strategie. Ein Maß für die Güte eines Algorithmus ist die Anzahl der Cache-Fehler. Zeige also, dass es für alle $\xi \in \mathbb{N}$ ein $\vec{a} = (a_1, \ldots, a_m) \in [5]^m$ gibt, sodass

$$error_{look-ahead-3} - error_{standard} \geq \xi,$$

wobei $error_{look-ahead-3}$ die Anzahl der Cache-Fehler des Look-Ahead-3 und $error_{standard}$ die Anzahl der Cache-Fehler des Standardalgorithmus und angibt.

Aufgabe 3.3 (4 Punkte)
Betrachte den nachfolgenden Algorithmus mit Eingabeparameter $n \in \mathbb{N}$ und Ausgabeparameter x.

```
1 x ← 0
2 WHILE n>0 DO
3           z ← n MOD 10
4           n ← n - z
5           x ← x + z
6           n ← n / 10
7 END
```

Was berechnet dieser Algorithmus? Zeige durch die Wahl einer geeigneten Invarianten die Korrektheit deiner Vermutung.

Aufgabe 3.4 (4 Punkte)
Es stehen zwei Stacks zur Verfügung. Die Ausgangssituation für den nachfolgenden Algorithmus bei Eingabeparameter $n \in \mathbb{N}_0$ ist: Einer der beiden Stacks ist leer und der andere enthält mindestens ein Element.

Arbeitsweise des Algorithmus:

1. Ist $n > 0$, versuche n Elemente des gefüllten Stacks auf den leeren Stack zu schichten. Sei $y \in \mathbb{N}$ die Anzahl der Elemente im gefüllten Stack.

   Fall a: Sind nicht genügend Elemente vorhanden, d.h. $n > y$, rufe nach dem Umschichten der ersten y Elemente den Algorithmus mit dem Wert $n - y$ auf.

   Fall b: Sind genügend Elemente vorhanden, d.h. $n \leq y$, zähle nach dem Umschichten die verbliebenen, nicht verschobenen Elemente durch POP()-Aufrufe und rufe den Algorithmus anschließend für diesen Wert auf.

2. Ist $n = 0$, stoppe.

Zeige, dass $\mathcal{O}(nx + x^2)$ eine obere Schranke für die Laufzeit des Algorithmus ist, falls dieser mit $n \in \mathbb{N}$ aufgerufen wird und sich vor dem Aufruf in dem nicht leeren Stack $x > 0$ Elemente befinden.