Datenstrukturen: Bag (Multimenge) **Prüfungsvorbereitung**

Aufgabe 1

Welche strukturellen Merkmale hat die Datenstruktur *Bag*?

Aufgabe 2

Beschreibe die typischen Operationen der Datenstruktur *Bag*.

Aufgabe 3

Gib zwei unterschiedliche Anwendungen der Datenstruktur *Bag* an.

Aufgabe 4

Das folgende Speicherabbild enthält einen *Bag*. Der Einstiegspunkt (oder Kopf) dieser Datenstruktur befindet sich an der Adresse `16`. Wie sieht der Speicher aus, nachdem das Element `04` aus dem Bag gelöscht wurde? Beschreibe die dafür nötigen Änderungen in den Zellen.

	0	1	2	3	4	5	6	7	8	9
0		03	00	15	01	04	03	14	08	05
1	05	15	09	10	07	06	08	12	02	19

- Daten und Adressen bestehen jeweils aus einer Zahl zwischen 0 und 19.
- Ein Knoten (*Node*) belegt zwei aufeinanderfolgenden Speicherzellen; die erste enthält einen Datenwert und die zweite einen Adresswert.
- Beachte, dass der Speicher logisch aus einer einzigen langen Liste von Speicherzellen besteht und die Speicherzelle mit der Adresse `10` unmittelbar auf die Speicherzelle mit der Adresse `09` folgt.
- Der Speicher enthält auch noch andere oder vewaiste Daten.

Aufgabe 5

Welche Ausgabe macht der folgende Code?

```
1  def gf(a, q, n):
2      for k in range(0, n):
3          yield a * q**k
4
5  for x in gf(3, 2, 4):
6      print(x)
```

Aufgabe 6

Implementiere die Methode `add(item)` in der Klasse `Bag` auf der Grundlage des folgenden Codes.

```
class Node:

    def __init__(self, item, nxt):
        self.item = item # Nutzdaten
        self.nxt = nxt   # Zeiger

class Bag:

    def __init__(self):
        self.first = None # Zeiger auf 1. Knoten
        self.n = 0        # Anzahl Elemente

    # hier die Methode implementieren
```

Aufgabe 7

Welche asymptotische Laufzeiten haben die folgenden Methoden für einen *Bag* mit n Elementen, wenn diese Datenstruktur auf der Grundlage einer (einfach) verketteten Liste bestmöglich implementiert wurde?

- `Bag()`
- `add(item)`
- `__iter__()`
- `clear()`
- `size()`

Aufgabe ?

Welche Ausgabe macht der folgende Code?

```
def af(a, d, n):
    for k in range(0, n):
        yield a + k*d

```

```
for x in af(17, -3, 5):
    print(x)
```

Aufgabe ?

Löse die Aufgaben unterhalb des folgenden Python-Codes.

```
class Node:

    def __init__(self, item, nxt):
        self.item = item # Nutzdaten
        self.nxt = nxt   # Zeiger

class Bag:

    def __init__(self):
        self.first = None # Zeiger auf 1. Knoten
        self.n = 0        # Anzahl Elemente

    def __iter__(self):
        actual_node = self.first
        while actual_node != None:
            yield actual_node.item
```

(a) Definiere auf dem Beiblatt eine syntaktisch korrekte Methode `clear(...)`, so dass `b.clear()` alle Elemente effizient aus einem Objekt `b` der Klasse `Bag` „entfernt“.

(b) Definiere auf dem Beiblatt eine syntaktisch korrekte Methode `add(...)`, so dass `b.add(item)` ein Element `item` effizient einem Objekt `b` der Klasse `Bag` hinzufügt.

(c) Welche Ausgabe macht der folgende Code? Gehe davon aus, dass Teil (b) korrekt gelöst wurde.

```
b = Bag()
b.add(7)
b.add(3)
b.add(9)
b.add(5)

for x in b:
    print(x)
```