Aufgabe 2

Gegeben sei die folgende Java-Implementierung einer doppelt-verketteten Liste.

```java
class DoubleLinkedList {
  private Item head;

  public DoubleLinkedList() {
    head = null;
  }

  public Item append(Object val) {
    if (head == null) {
      head = new Item(val, null, null);
      head.prev = head;
      head.next = head;
    } else {
      Item item = new Item(val, head.prev, head);
      head.prev.next = item;
      head.prev = item;
    }
    return head.prev;
  }

  public Item search(Object val) {
    // ...
  }

  public void delete(Object val) {
    // ...
  }

  class Item {
    private Object val;
    private Item prev;
    private Item next;

    public Item(Object val, Item prev, Item next) {
      this.val = val;
      this.prev = prev;
      this.next = next;
    }
  }
}
```

(a) Skizzieren Sie den Zustand der Datenstruktur nach Aufruf der folgenden Befehlssequenz. Um Variablen mit Zeigern auf Objekte darzustellen, können Sie mit dem Variablennamen beschriftete Pfeile verwenden.

```java
DoubleLinkedList list = new DoubleLinkedList();
list.append("a");
list.append("b");
list.append("c");
```

(b) Implementieren Sie in der Klasse `DoubleLinkedList` die Methode `search`, die zu einem gegebenen Wert das Item der Liste mit dem entsprechenden Wert, oder `null` falls der Wert nicht in der Liste enthalten ist, zurückgibt.

```
23    public Item search(Object val) {
24      Item item = null;
25      if (head != null) {
26        item = head;
27        do {
28          if (item.val == val) {
29            return item;
30          }
31          item = item.next;
32        } while (!item.equals(head));
33      }
34      return null;
35    }
```

github: raw

(c) Implementieren Sie in der Klasse `DoubleLinkedList` die Methode `delete`, die das erste Vorkommen eines Wertes aus der Liste entfernt. Ist der Wert nicht in der Liste enthalten, terminiert die Methode „stillschweigend“, d. h. ohne Änderung der Liste und ohne Fehlermeldung. Sie dürfen die Methode `search` aus Teilaufgabe b) verwenden, auch wenn Sie sie nicht implementiert haben.

```
37    public void delete(Object val) {
38      Item item = search(val);
39      if (item != null) {
40        if (head.next.equals(head)) {
41          head = null;
42        } else {
43          if (item.equals(head)) {
44            head = item.next;
45          }
46          item.prev.next = item.next;
47          item.next.prev = item.prev;
48        }
49      }
50    }
```

github: raw

(d) Beschreiben Sie die notwendigen Änderungen an der Datenstruktur und an den bisherigen Implementierungen, um eine Methode `size`, die die Anzahl der enthaltenen Items zurück gibt, mit Laufzeit $\mathcal{O}(1)$ zu realisieren.

In der Klasse wird ein Zähler eingefügt, der bei jedem Aufruf der Methode `append` eines nach oben gezählt wird und bei jedem erfolgreichen Löschen in der delete-Methode eines nach unten gezählt wird. Mit return kann der Zählerstand in $\mathcal{O}(1)$ ausgegeben werden. Dazu müsste ein Getter zum Ausgeben implementiert werden. Die Datenstruktur bleibt unverändert.