1) Lineare Liste

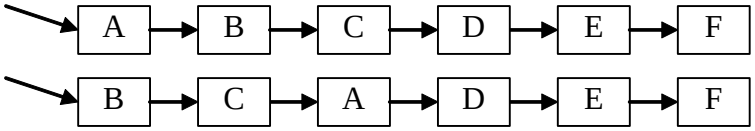
In einer linearen Liste soll das erste Element um k Stellen in der Liste nach hinten verschoben werden (move). Kann nicht um k Elemente verschoben werden, soll das Element hinten angehängt werden.

Verwenden Sie dazu folgende Definition der Liste:

```
class List {
  Node head;
  void move(int k) {}
}
```

```
class Node {
   Node next;
   String value;
}
```

Beispiel: `aList.move(2);`



2) Binärer Suchbaum

Ein Kaufhaus hat eine Vielzahl von Artikeln im Angebot. Diese Artikel haben einen bestimmten Preis und eine Beschreibung, die in folgender Datenstruktur verwaltet werden:

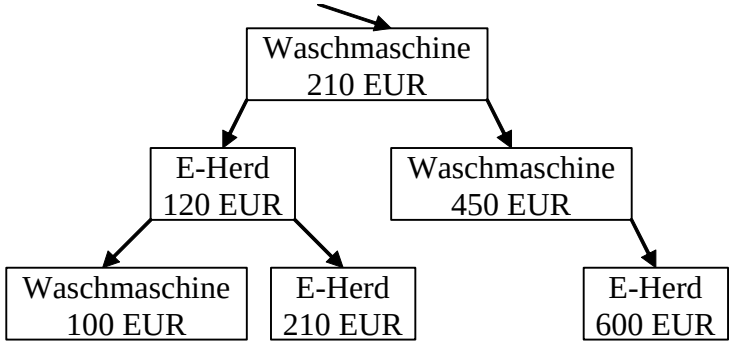
```
class ItemNode { // Ein Artikel
  ItemNode left, right;
  String description;
  double price; // Preis des Artikels
  void printItem() //Ausgabe des Artikels (description + price)
}
```

Die Artikel werden in einem binären Suchbaum (sortiert nach dem Preis) verwaltet:

```
class InventoryTree {
  ItemNode root;
  void printCheapest(double maxPrice) {...}
}
```

Das Kaufhaus möchte nun sein Image als Ramsch-Laden loswerden, weshalb alle billigen Artikel abtransportiert werden sollen. Gesucht ist die Implementierung für die angegebene Methode `printCheapest`, die die billigsten Artikel sortiert bis zu einem bestimmten Preis ausgibt.

Beispiel:
`printCheapest(200)`
liefert bei nebenstehendem Baum als Ergebnis:
Waschmaschine, 100 EUR
E-Herd, 120 EUR



Anm.: Achten Sie auf eine effiziente Implementierung (es sollen möglichst wenige Knoten im Baum besucht werden). Nutzen Sie dazu die Sortierung des Suchbaums aus.

3) Hashtabelle

Gegeben ist die folgende Schnittstelle einer Hashtabelle:

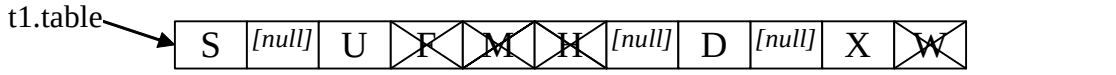
```
public class Hashtable {
     Node[] table;
     int nOfElements, nOfDeleted;

     public Hashtable minus(Hashtable t) {...}

     public boolean isEqualTo(Hashtable t) {...}

// Weitere Methoden (zur freien Verwendung)
     public Hashtable(int initSize) {...}
     public void insert(String val) {...}
     public void delete(String val) {...}
     public boolean contains(String val) {...}
}
```

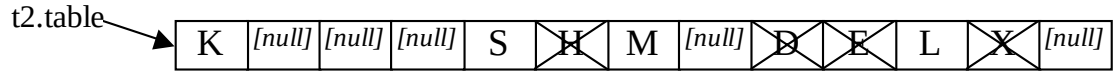
```
class Node {
    boolean deleted;
    String value;

    Node(String val) {
       value = val;
    }
}
```

Implementieren Sie die zwei Methoden `minus` und `isEqualTo`:

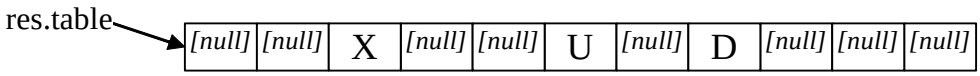
- **`minus`**`(Hashtable t)` berechnet die Differenz der in beiden Hashtabellen enthaltenen (nicht gelöschten) Elemente (alle Elemente, die in der ersten, aber nicht in der zweiten Hashtabelle enthalten sind). Das Resultat soll als neue Hashtabelle zurückgegeben werden.
- **`isEqualTo`**`(Hashtable t)` vergleicht den Inhalt der beiden Hashtabellen (true, wenn die gleichen Elemente enthalten sind).

Beispiel:





res = t1.minus(t2);



/ res.isEqualTo(t1) == false */* res.insert("S"); */* res.isEqualTo(t1) == true */*