CHRISTIAN-ALBRECHTS-UNIVERSITÄT ZU KIEL
Institut für Informatik, Arbeitsgruppe Algorithmen und Komplexität
Prof. Dr. K. Jansen, K.-M. Klein, F. Land M. Rau

26. Mai 2016

Präsenzaufgaben zur Vorlesung »Algorithmen und Datenstrukturen«

Blatt 7

Präsenzaufgabe 7.1 (Hashing)
Konstruieren Sie die Hashtabelle der Größe $m = 23$, die durch Einfügen der Elemente

```
47 17 24 70 22 01 40 45 36 59
```

mit der

1. Divisionsmethode $h(x) = x \mod m$,
2. Multiplikationsmethode $h(x) = \lfloor (xc \mod 1)m \rfloor = \lfloor (xc - \lfloor xc \rfloor)m \rfloor$ mit $c = 1/2$ und
3. erweiterter Divisionsmethode $h(x) = (ax+b) \mod m$, mit $a = 5$ und $b = 3$

entsteht.

Präsenzaufgabe 7.2 (Hashing)
Implementieren Sie eine Hash-Tabelle für Hashing mit direkter Verkettung in Java. Verwenden Sie als Hash-Funktion die erweiterte Divisionsmethode.

Nutzen Sie für eine Implementierung in Java die Methodendeklarationen auf der Rückseite. Implementieren Sie entsprechend die auf der Rückseite beschriebenen Operationen *hash*, *lookup*, *insert* und *delete*.

Präsenzaufgabe 7.3 (Suchbäume)
Fügen Sie die Elemente $6, 4, 2, 5, 3, 1$ in dieser Reihenfolge in einen initial leeren Suchbaum ein. Entfernen Sie dann die Elemente $4, 2, 5$ (in dieser Reihenfolge). Geben Sie Zwischenschritte an.

```
//HashTable for keys that are strictly positive integers.
public class HashTable {

    static final int DEFAULT_M=101;
    private static final int EMPTY=0;
    private static final int DELETED=-1;

    private Node[] t;
    private int m;
    //Definiert die Parameter der Hashfunktion
    private final int a=13, b=3;

    //Konstruktoren
    public HashTable(){
        this(DEFAULT_M);
    }

    public HashTable(int size){
        m=size;
        t = new Node[m];
    }

   // Gibt den Wert der Hash-Funktion zurueck
    private int hash(int x){}


   // Ueberpruefe, ob x in der Hashtabelle vorhanden ist.
    public boolean lookup(int x){}


    // Fuegt ein Element x in die Tabelle ein. Wirft eine
       Exception, wenn x bereits vorhanden ist oder kein
       freier Platz vorhanden ist.
    public void insert(int x) throws HashTableException {}


    //Loescht das Element x aus der Hashtabelle. Wirft eine
       Exception, wenn das Element nicht vorhanden ist.
    public void delete(int x) throws HashTableException {}

    private class Node {
        public int element;
        public Node next;

        public Node(int elem, Node next){
            this.element = elem;
            this.next = next;
        }
    }
}
```