Übungsblatt 12

Aufgabe 1 - Mustererkennung

- Gebe die Zustandsüberführungsfunktion für einen *Moore-Automaten* an, der in einer Folge von Zeichen aus Σ mit $\{a,b,c\} \subseteq \Sigma$ das Muster `ababaca` erkennt. Benutze die aus der Vorlesung bekannte tabellarische Notation.
- Gebe die Zustandsüberführungsfunktion für einen *Moore-Automaten* an, der in einer Folge von Zeichen aus Σ mit $\{a,b,c\} \subseteq \Sigma$ das Muster `aabcaac` erkennt. Benutze die aus der Vorlesung bekannte tabellarische Notation.

Aufgabe 2 - Hashing

In der Vorlesung ist Hashing als eine Form zum schnellen Ablegen und Wiederfinden von Informationen präsentiert worden.
Die dort vorgestellte Inplementierung benutzt Listen, um Objekte mit kollidierendem Hashcode abzulegen.
Die Implementierung kann aber auch ohne Listen erfolgen:
Tritt eine Kollision auf, wird für das neu abzulegende Objekt das "nächste" freie Element des Feldes verwendet – also der kleinste größere Index, dessen zugehöriges Element auf `null` verweist. Wird bei der Suche nach so einem Element das Ende des Feldes erreicht, so wird mit dem Index `0` fortgefahren.
Überlege Dir, wie die Methode `contains` arbeiten muss.
Welches besondere Problem wird bei dieser Implementierung durch das Löschen eines Wertes hervorgerufen?
Überlege Dir eine Lösung!
Implementiere die Klasse `SimpleHashTable` mit den bekannten Methoden:

```
public void put( T o )
public boolean contains( T o )
public void remove( T o )
public int size()
private void rehash()
```

Aufgabe 3 - Heap

- Wie viele Ebenen hat ein Heap mit
  – 32 000 Elementen
  – 1 000 000 Elementen?
  An welchen Stellen der Algorithmen zum Einfügen und Löschen beeinflusst die Zahl der Ebenen die Laufzeit?

- Das unten stehende Feld bildet einen Heap. Gib die zugehörige Baumdarstellung an!

	0	1	2	3	4	5	6	7	8
pq		22	15	17	12	14	9	11	4

- Erzeuge aus dem Heap `pq` mit Hilfe der folgenden Anweisungen (siehe Folie 1185) eine aufsteigende Sortierung:

```
for ( int i = elements.length-1; i >= 0; i-- ) {
    elements[i] = pq.poll();
}
```

  Gib an, wie die beiden Felder am Ende jedes Schleifendurchlaufs aussehen

	0	1	2	3	4	5	6	7	8
pq		22	15	17	12	14	9	11	4

	0	1	2	3	4	5	6	7
elements	0	0	0	0	0	0	0	0

- Die in der Vorlesung vorgestellte Methode `poll` entfernt den Wert in der Wurzel des Heaps, ersetzt ihn durch einen neuen Wert und stellt den Heap mit der Methode `heapify` aus den beiden Teilbäumen, die Heaps sind, wieder her. Dieses Vorgehen kann auch benutzt werden, um aus einer Folge von unstrukturierten Werten einen Heap herzustellen.

  Stelle das folgende Feld als Baum gemäß der für Heaps gültigen Abbildung der Werte dar: Es bildet dann **keinen** Heap! Füge nun ausgehend von den Vorgängerknoten der Blätter – ein einzelner Knoten ist ein Heap – analog zu `heapify` – ebenenweise von unten immer größere Heaps zusammen.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	5	13	4	17	12	10	34	15	7	9	16	1	23	22

  Beschreibe das so entwickelte Vorgehen durch eine allgemeine Handlungsanweisung!