Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2019

66115

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach:	**Informatik (vertieft studiert)**
Einzelprüfung:	**Theoretische Informatik, Algorithmen**
Anzahl der gestellten Themen (Aufgaben):	2
Anzahl der Druckseiten dieser Vorlage:	12

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Aufgabe 1 (Reguläre Sprachen) [36 PUNKTE]

(a) [10 Punkte] Betrachten Sie den regulären Ausdruck

$$\alpha = \big((a+b)^*ab\big) + \big((a+b)^*bb\big) \; .$$

Geben Sie einen minimalen DEA für die von diesem Ausdruck beschriebene Sprache $L(\alpha)$ an.

(b) [14 Punkte] Betrachten Sie die folgenden Sprachen über dem Alphabet $\Sigma = \{a, b\}$:

$$L_1 = \{w \in \{a,b\}^* \mid \text{es existieren } u, v \in \{a,b\}^* \text{ mit } |u| = |v| \text{ und } w = uav\}$$
$$L_2 = \{w \in \{a\}^* \mid \text{es existieren } u, v \in \{a\}^* \text{ mit } |u| = |v| \text{ und } w = uav \}$$

Geben Sie für die Sprachen jeweils an, ob sie regulär sind oder nicht, und beweisen Sie Ihre Antwort. Für den Beweis von Regularität reicht die Angabe eines Automaten oder regulären Ausdrucks für die jeweilige Sprache.

(c) [12 Punkte] Wir definieren die Menge der Permutationen eines Wortes w. Sei $w = a_1 \cdots a_n$ mit $a_i \in \Sigma$ für alle $i = 1, \ldots, n$. Dann ist

$$\text{Perm}(w) := \{a_{\pi(1)} \cdots a_{\pi(n)} \mid \pi \text{ ist eine Permutation von } \{1, \ldots, n\}\} \; .$$

So ist z.B. $\text{Perm}(aab) = \{aab, aba, baa\}$. Weiter definieren wir $\text{Perm}(L)$ für eine Sprache L als $\bigcup_{w \in L} \text{Perm}(w)$.

Zeigen oder widerlegen Sie: Falls L eine reguläre Sprache ist, dann ist auch $\text{Perm}(L)$ eine reguläre Sprache.

Aufgabe 2 (Kontextfreie Sprachen) [24 PUNKTE]

(a) [10 Punkte] Geben Sie einen (deterministischen oder nichtdeterministischen) Kellerautomaten für die Sprache

$$\{a^i b^j c^k \mid i, j, k \in \mathbb{N} \text{ und } i + k < j\}$$

an.

(b) [14 Punkte] Sei

$$L_3 = \{wc^i w \mid w \in \{a,b\}^* \text{ und } i \geq 1\}$$

eine Sprache über dem Alphabet $\{a, b, c\}$. Zeigen Sie, dass L_3 nicht kontextfrei ist.

Fortsetzung nächste Seite!

Aufgabe 3 (Entscheidbarkeit) [30 PUNKTE]

(a) [18 Punkte] Wir betrachten eine Gödelisierung von Turingmaschinen und bezeichnen mit M_w die Turingmaschine, die gemäß dieser Kodierung vom Binärwort w kodiert wird. Außerdem bezeichnen wir mit $M_w(x)$ die Berechnung der Maschine M_w bei Eingabe x. Betrachten Sie die Sprache

$$L_4 = \{w \mid M_w(0) \text{ hält nach weniger Schritten an als } M_w(1)\}$$

Hier sind Wörter w bei denen sowohl $M_w(0)$ als auch $M_w(1)$ nicht anhalten, nicht in L. Beweisen Sie, dass L_4 unentscheidbar ist.

(b) [12 Punkte] Das CYK-Verfahren ist ein Algorithmus, der für eine bestimmte kontextfreie Grammatik G in polynomieller Zeit entscheidet, ob ein gegebenes Wort w in $L(G)$ liegt. Somit ist jede kontextfreie Sprache entscheidbar. Zeigen oder widerlegen Sie folgende Aussage:

Seien K_1, K_2 und K_3 kontextfreie Sprachen. Sei L_5 die Sprache, für die $L_5 = (K_1 \cap K_2) \setminus K_3$ gilt. Dann ist L_5 entscheidbar.

Aufgabe 4 (Komplexität) [30 PUNKTE]

Betrachten Sie die folgenden Probleme:

SAT	
Gegeben:	Aussagenlogische Formel F in KNF.
Frage:	Gibt es mindestens eine erfüllende Belegung für F?

DOPPELSAT	
Gegeben:	Aussagenlogische Formel F in KNF.
Frage:	Gibt es mindestens eine erfüllende Belegung für F, in der mindestens zwei Literale pro Klausel wahr sind?

(a) [24 Punkte] Führen Sie eine polynomielle Reduktion von SAT auf DOPPELSAT durch.

(b) [6 Punkte] Zeigen Sie, dass DOPPELSAT NP-vollständig ist.

Fortsetzung nächste Seite!

Aufgabe 5 (Sortierverfahren) [40 PUNKTE]

In der folgenden Aufgabe soll ein Feld A von ganzen Zahlen aufsteigend sortiert werden. Das Feld habe n Elemente A[0] bis A[n−1]. Der folgende Algorithmus (in der Notation des Informatik-Duden) sei gegeben:

```
1  procedure quicksort(links, rechts : integer)
2  var i, j, x : integer;
3  begin
4    i := links;
5    j := rechts;
6    if j > i then begin
7      x := A[links];
8      repeat
9        while A[i] < x do i := i+1;
10       while A[j] > x do j := j−1;
11       if i ≤ j then begin
12         tmp := A[i]; A[i] := A[j]; A[j] := tmp;
13         i := i+1; j := j−1;
14       end
15     until i > j;
16     quicksort(links, j);
17     quicksort(i, rechts);
18   end
19 end
```

Der initiale Aufruf der Prozedur lautet:

quicksort(0,n−1)

(a) [18 Punkte] Sortieren Sie das folgende Feld der Länge 7 mittels des Algorithmus. Notieren Sie jeweils alle Aufrufe der Prozedur quicksort mit den konkreten Parameterwerten. Geben Sie zudem für jeden Aufruf der Prozedur den Wert des in Zeile 7 gewählten Elements an.

Index	0	1	2	3	4	5	6
Wert	27	32	3	6	17	44	42

(b) [5 Punkte] Angenommen, die Bedingung j > i in Zeile 6 des Algorithmus wird ersetzt durch die Bedingung j ≥ i. Ist der Algorithmus weiterhin korrekt? Begründen Sie Ihre Antwort.

(c) [5 Punkte] Angenommen, die Bedingung i ≤ j in Zeile 11 des Algorithmus wird ersetzt durch die Bedingung i < j. Ist der Algorithmus weiterhin korrekt? Begründen Sie Ihre Antwort.

(d) [5 Punkte] Wie muss das Feld A gestaltet sein, damit der Algorithmus mit der geringsten Anzahl von Schritten terminiert? Betrachten Sie dazu vor allem Zeile 7. Begründen Sie Ihre Antwort und geben Sie ein Beispiel.

Fortsetzung nächste Seite!

(e) [7 Punkte] Die rekursiven Aufrufe in den Zeilen 16 und 17 des Algorithmus werden zur Laufzeit des Computers auf dem Stack verwaltet. Die Anzahl der Aufrufe von quicksort auf dem Stack abhängig von der Eingabegröße n sei mit $s(n)$ bezeichnet. Geben Sie die Komplexitätsklasse von $s(n)$ für den schlimmsten möglichen Fall an. Begründen Sie Ihre Antwort.

Aufgabe 6 (O-Notation) [40 PUNKTE]

(a) [8 Punkte] Sortieren Sie die unten angegebenen Funktionen der O-Klassen $O(a(n))$, $O(b(n))$, $O(c(n))$, $O(d(n))$ und $O(e(n))$ bezüglich ihrer Teilmengenbeziehungen. Nutzen Sie ausschließlich die echte Teilmenge $\subset$ sowie die Gleichheit $=$ für die Beziehung zwischen den Mengen. Folgendes Beispiel illustriert diese Schreibweise für einige Funktionen f_1 bis f_5 (diese haben nichts mit den unten angegebenen Funktionen zu tun):

$$O(f_4(n)) \subset O(f_3(n)) = O(f_5(n)) \subset O(f_1(n)) = O(f_2(n))$$

Die angegebenen Beziehungen müssen weder bewiesen noch begründet werden.

- $a(n) = n^2 \cdot \log_2(n) + 42$
- $b(n) = 2^n + n^4$
- $c(n) = 2^{2 \cdot n}$
- $d(n) = 2^{n+3}$
- $e(n) = \sqrt{n^5}$

(b) Beweisen Sie die folgenden Aussagen formal nach den Definitionen der O-Notation oder widerlegen Sie sie.

(i) [8 Punkte] $O(n \cdot \log_2 n) \subseteq O(n \cdot (\log_2 n)^2)$

(ii) [8 Punkte] $2^{n+1} \in O(2^n)$

(c) Bestimmen Sie eine asymptotische Lösung (in Θ-Schreibweise) für die folgende Rekursionsgleichung:

(i) [8 Punkte] $T(n) = 4 \cdot T(\frac{n}{2}) + n^2$

(ii) [8 Punkte] $T(n) = T(\frac{n}{2}) + \frac{1}{2}n^2 + n$

Fortsetzung nächste Seite!

Aufgabe 7 (Bäume) [40 PUNKTE]

Gegeben sei die folgende Realisierung von binären Bäumen (in einer an Java angelehnten Notation):

```
1 class Node {
2   Node left, right;
3   int value;
4 }
```

(a) [8 Punkte] Beschreiben Sie in möglichst wenigen Worten, was die folgende Methode foo auf einem nicht-leeren binären Baum berechnet.

```
1  int foo( Node node ){
2    int b = node.value;
3    if ( b < 0 ) {
4      b = -1 * b;
5    }
6    if ( node.left != null ) {
7      b = b + foo( node.left );
8    }
9    if ( node.right != null ) {
10     b = b + foo( node.right );
11   }
12   return b;
13 }
```

(b) [6 Punkte] Die Laufzeit der Methode foo(tree) ist linear in n, der Anzahl von Knoten im übergebenen Baum tree. Begründen Sie kurz, warum foo(tree) eine lineare Laufzeit hat.

(c) [14 Punkte] Betrachten Sie den folgenden Algorithmus für nicht-leere, binäre Bäume. Beschreiben Sie in möglichst wenigen Worten die Wirkung der Methode magic(tree). Welche Rolle spielt dabei die Methode max?

```
1  void magic( Node node ){
2    Node m = max ( node );
3    if (m.value > node.value ) {
4      // Werte von m und node vertauschen
5      int tmp = m.value;
6      m.value = node.value;
7      node.value = tmp;
8    }
9    if (node.left != null )
10     magic( node.left );
11   if (node.right != null )
12     magic( node.right );
13 }
14 Node max( Node node ){
15   Node max = node;
16   if ( node.left != null ){
```

Fortsetzung nächste Seite!

```
17      Node tmp = max( node.left );
18      if (tmp.value > max.value ) max = tmp ;
19    }
20    if ( node.right != null ){
21      Node tmp = max( node.right );
22      if (tmp.value > max.value ) max = tmp ;
23    }
24    return max ;
25  }
```

(d) [12 Punkte] Geben Sie in Abhängigkeit von n, der Anzahl von Knoten im übergebenen Baum tree, jeweils eine Rekursionsgleichung für die asymptotische Best-Case-Laufzeit ($B(n)$) und Worst-Case-Laufzeit ($W(n)$) des Aufrufs magic(tree) sowie die entsprechende Komplexitätsklasse (Θ) an. Begründen Sie Ihre Antwort.

Hinweis: Überlegen Sie, ob die Struktur des übergebenen Baumes Einfluss auf die Laufzeit hat. Die lineare Laufzeit von max(t) in der Anzahl der Knoten des Baumes t darf vorausgesetzt werden.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Aufgabe 1 (Turing-Maschine) [26 PUNKTE]

Gesucht ist eine Turing-Maschine mit genau einem beidseitig unendlichen Band, die die Funktion $f : \mathbb{N} \to \mathbb{N}$ mit $f(x) = 3x$ berechnet. Zu Beginn der Berechnung steht die Eingabe binär codiert auf dem Band, wobei der Kopf auf die linkeste Ziffer (most significant bit) zeigt. Am Ende der Berechnung soll der Funktionswert binär codiert auf dem Band stehen, wobei der Kopf auf ein beliebiges Feld zeigen darf.

(a) Beschreiben Sie zunächst in Worten die Arbeitsweise Ihrer Maschine.

(b) Geben Sie dann das kommentierte Programm der Turing-Maschine an und erklären Sie die Bedeutung der verwendeten Zustände.

Aufgabe 2 (Zwei Abschlusseigenschaften von REG und CFL) [24 PUNKTE]

Es sei $\Sigma = \{a, b\}$ das Alphabet. Die folgenden Aussagen A und B werden betrachtet.

A: Für jede reguläre Sprache $L_1 \subseteq \Sigma^*$ ist auch die Sprache L_1' regulär, wobei $L_1' = \{w \in \Sigma^* \mid \text{für alle Wörter } w' \in \Sigma^* \text{ gilt } ww' \in L_1\}$.

B: Für jede kontextfreie Sprache $L_2 \subseteq \Sigma^*$ ist auch die Sprache L_2' kontextfrei, wobei $L_2' = \{w \in \{a, b, \#\}^* \mid \exists n \geq 1 \, \exists u_1, \ldots, u_n, v_1, \ldots, v_n \in L_2 \text{ mit } w = u_1 \cdots u_n \# v_1 \cdots v_n\}$. D.h. L_2' besteht aus den Wörtern $u\#v$, sodass sich u und v aus der gleichen Anzahl von Wörtern aus L_2 bilden lassen.

(a) Zeigen Sie die Aussage A. Es genügt die Beschreibung der Konstruktion, die aus einem endlichen Automaten für die Sprache L_1 einen endlichen Automaten für die Sprache L_1' erzeugt.

(b) Zeigen Sie die Aussage B. Es genügt die Beschreibung der Konstruktion, die aus einer kontextfreien Grammatik für die Sprache L_2 eine kontextfreie Grammatik für die Sprache L_2' erzeugt.

Aufgabe 3 (Einordnung von Mengen in Komplexitätsklassen) [42 PUNKTE]

Sei $M_0, M_1, \ldots$ eine Gödelisierung der Registermaschinen (RAMs). Weiter bezeichne $\text{bin}(x)$ die Binärdarstellung der Zahl $x \in \mathbb{N}$. Wir betrachten folgende Klassen von Sprachen über dem Alphabet $\Sigma = \{0, 1, \#\}$.

$$
\begin{aligned}
\text{REG} &= \{L \subseteq \Sigma^* \mid L \text{ ist regulär}\} \\
\text{P} &= \{L \subseteq \Sigma^* \mid L \text{ ist in deterministischer Polynomialzeit entscheidbar}\} \\
\text{NP} &= \{L \subseteq \Sigma^* \mid L \text{ wird von einer nichtdeterministischen Polynomialzeit-Maschine akzeptiert}\} \\
\text{REC} &= \{L \subseteq \Sigma^* \mid L \text{ ist entscheidbar}\} \\
\text{RE} &= \{L \subseteq \Sigma^* \mid L \text{ ist aufzählbar}\} \\
\text{ALL} &= \{L \mid L \subseteq \Sigma^*\}
\end{aligned}
$$

Fortsetzung nächste Seite!

Es gilt

$$\text{REG} \subsetneq \text{P} \subseteq \text{NP} \subsetneq \text{REC} \subsetneq \text{RE} \subsetneq \text{ALL}. \qquad (*)$$

Weiter seien folgende Sprachen über dem Alphabet $\Sigma = \{0, 1, \#\}$ gegeben.

$L_1 = \{w \in \Sigma^* \mid \exists n \in \mathbb{N}, w = 0^n 1^n 0^n\}$

$L_2 = \{\text{bin}(x) \mid x \in \mathbb{N} \text{ und } M_x \text{ hält nicht bei Eingabe } x\}$

$L_3 = \{\text{bin}(x) \mid x \in \mathbb{N} \text{ und } x \text{ ist durch 4 teilbar}\}$

$L_4 = \{\text{bin}(x) \mid x \in \mathbb{N} \text{ und die von } M_x \text{ berechnete Funktion } \mathbb{N} \to \mathbb{N} \text{ ist nicht injektiv}\}$

$L_5 = \{\text{bin}(a_1)\#\text{bin}(a_2)\#\cdots\#\text{bin}(a_n)\#\text{bin}(b) \mid a_1, \ldots, a_n, b \in \mathbb{N} \text{ und } \exists I \subseteq \{1, \ldots, n\}, \sum_{i \in I} a_i = b\}$

Geben Sie für jedes L_i Folgendes an:

(a) die (nach aktuellem Kenntnisstand) kleinste Klasse $\mathcal{C}$ aus der Inklusionskette $(*)$ mit der Eigenschaft $L_i \in \mathcal{C}$;

(b) eine kurze Begründung für $L_i \in \mathcal{C}$ (ca. 2-3 Zeilen Begründung, kein ausführlicher Beweis);

(c) eine kurze Begründung dafür, dass L_i (nach aktuellem Kenntnisstand) nicht in der nächst kleineren Klasse liegt (ca. 2-3 Zeilen Begründung, kein ausführlicher Beweis).

Aufgabe 4 (Reduktion von 3-FARB auf 7-FARB) [28 PUNKTE]

Im Folgenden werden endliche, ungerichtete Graphen betrachtet, wobei ein endlicher ungerichteter Graph G ein Paar (V, E) ist, sodass

- V eine nichtleere, endliche Menge ist (die Knotenmenge) und
- $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$ (die Kantenmenge), d.h. E ist eine Menge zweielementiger Teilmengen von V.

Für $k \geq 3$ betrachten wir das Problem

$$k\text{-FARB} = \{G \mid G \text{ ist ein } k\text{-färbbarer ungerichteter Graph}\},$$

wobei ein Graph $G = (V, E)$ genau dann k-färbbar heißt, falls seine Knoten so mit k Farben eingefärbt werden können, dass alle durch eine Kante verbundenen Knoten verschiedenfarbig sind (d.h. es gibt ein totales $c : V \to \{1, \ldots, k\}$ mit $\forall \{u, v\} \in E$ gilt $c(u) \neq c(v)$).

Zeigen Sie, dass 3-FARB in polynomieller Zeit auf 7-FARB reduzierbar ist (d.h. 3-FARB $\leq$ 7-FARB). Gehen Sie dazu wie folgt vor:

(a) Geben Sie eine totale, in Polynomialzeit berechenbare Reduktionsfunktion f an.
(Es ist hier kein Nachweis der Totalität und Polynomialzeit-Berechenbarkeit von f gefordert.)

(b) Zeigen Sie: Falls G in 3-FARB, so ist $f(G)$ in 7-FARB.

(c) Zeigen Sie: Falls $f(G)$ in 7-FARB, so ist G in 3-FARB.

Fortsetzung nächste Seite!

Aufgabe 5 ($\mathcal{O}$ -Notation) [35 PUNKTE]

(a) [5 Punkte] Geben Sie eine formale Definition von $\mathcal{O}(f)$.

(b) [10 Punkte] Zeigen oder widerlegen Sie: $\mathcal{O}(\log(n!)) = O(n \log n)$.

(c) [20 Punkte] Zeigen oder widerlegen Sie jeweils für Funktionen

$$f(n) = 10^4 \cdot \sqrt{n} \text{ und } g(n) = \begin{cases} 10, & \text{für } n \bmod 2 = 0 \\ \frac{1}{2}n, & \text{sonst} \end{cases}$$

(i) $f \in \mathcal{O}(g)$.

(ii) $g \in \mathcal{O}(f)$.

Aufgabe 6 (Mastertheorem) [20 PUNKTE]

Der Hauptsatz der Laufzeitfunktionen ist bekanntlich folgendermaßen definiert:

$$\text{Sei } T(n) = \begin{cases} d \in \Theta(1), & \text{falls } n \leq k \\ a \cdot T(\frac{n}{b}) + g(n), & \text{sonst} \end{cases} \quad \text{mit } k \in \mathbb{N},\ a \geq 1 \text{ und } b > 1$$

Dann gilt

1. $g(n) \in \mathcal{O}(n^{\log_b a - \epsilon})$ für ein $\epsilon > 0$
   $\Rightarrow T(n) \in \Theta(n^{\log_b a})$
2. $g(n) \in \Theta(n^{\log_b a})$ $\Rightarrow T(n) \in \Theta(n^{log_b a} \cdot \log n) = \Theta(g(n) \cdot \log n)$
3. $g(n) \in \Omega(n^{\log_b a + \epsilon})$ für ein $\epsilon > 0$ und $a \cdot g(\frac{n}{b}) \leq c \cdot g(n)$ für fast alle n und ein c mit $0 < c < 1$
   $\Rightarrow T(n) \in \Theta(g(n))$

Bestimmen und begründen Sie formal mit Hilfe dieses Satzes welche Komplexität folgende Laufzeitfunktionen haben.

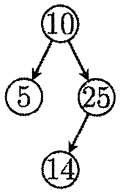
(a) [12 Punkte] $T(n) = 8T(\frac{n}{3}) + 5n^2$

(b) [8 Punkte] $T(n) = 9T(\frac{n}{3}) + 5n^2$

Fortsetzung nächste Seite!

Aufgabe 7 (AVL-Bäume) [32 PUNKTE]

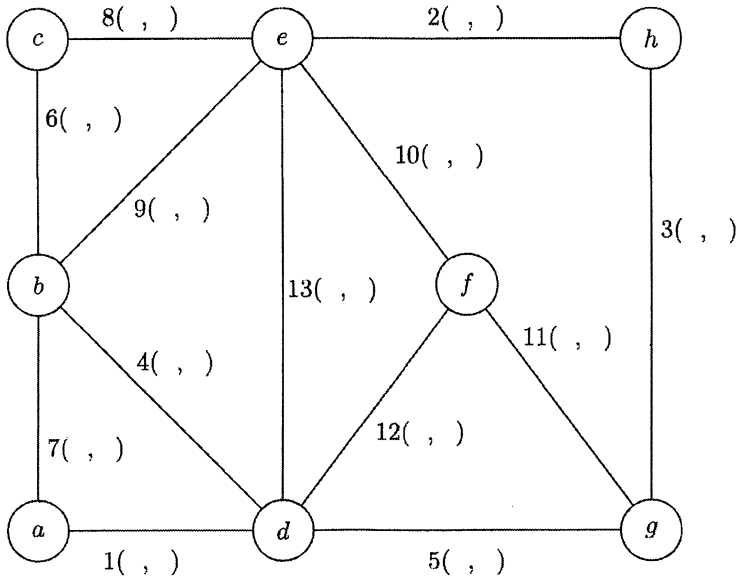
Fügen Sie (manuell) nacheinander die Zahlen 20, 31, 2, 17, 7 in folgenden AVL-Baum ein. Löschen Sie anschließend aus dem entstandenen Baum nacheinander 14 und 25.



Zeichnen Sie jeweils direkt nach jeder einzelnen Operation zum Einfügen oder Löschen eines Knotens, sowie nach jeder elementaren Rotation den entstehenden Baum. Insbesondere sind evtl. anfallende Doppelrotationen in zwei Schritten darzustellen. Geben Sie zudem an jedem Knoten die Balancewerte an.

Aufgabe 8 (Minimaler Spannbaum) [21 PUNKTE]

Gegeben Sei der folgende ungerichtete Graph mit Kantengewichten.



(a) [7 Punkte] Zeichnen Sie den (hier eindeutigen) minimalen Spannbaum.

(b) [14 Punkte] Geben Sie sowohl für den Algorithmus von Jarník-Prim als auch für den Algorithmus von Kruskal die Reihenfolge an, in der die Kanten hinzugefügt werden. Starten Sie für den Algorithmus von Jarník-Prim beim Knoten a.

Übernehmen Sie den Graph auf Ihre Bearbeitung und füllen Sie hierzu das Tupel jeder Kante e aus dem MST in der Form (n, m) aus, wobei die Kante e vom Algorithmus von Jarník-Prim als n'te Kante und vom Algorithmus von Kruskal als m'te Kante hinzugefügt wird. Lassen Sie andere Tupel unausgefüllt.

Fortsetzung nächste Seite!

Aufgabe 9 (Hashing) [12 PUNKTE]

Verwenden Sie die Hashfunktion $h(k,i) = (h'(k)+i^2) \bmod 11$ mit $h'(k) = k \bmod 13$, um die Werte 12, 29 und 17 in die folgende Hashtabelle einzufügen. Geben Sie zudem jeweils an auf welche Zellen der Hashtabelle zugegriffen wird.

0	1	2	3	4	5	6	7	8	9	10
			16		5				22	