Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr
2007

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: 5

Bitte wenden!

Thema Nr. 1

Aufgabe 1:

Gegeben sei das Alphabet $\sum = \{a, b\}$ und die Sprache $L_1 = \{waba | w \in \sum^*\}$

a) Zeigen Sie, dass L_1 regulär ist.

b) Geben Sie einen vollständigen deterministischen endlichen Automaten an, der die Sprache L_1 akzeptiert.

c) Konstruieren Sie einen minimalen deterministischen Automaten, der L_1 akzeptiert und weisen Sie dessen Minimalität nach.

Gegeben sei weiter die Sprache $L_2 = \left\{ww^R aba | w \in \sum^*\right\}$.

d) Stellen Sie eine geeignete Grammatik G auf, die die Sprache L_2 erzeugt, und geben Sie eine Ableitung der Wörter aba und aabbaaaba in G an.

e) Ist L_2 regulär bzw. kontextfrei? Begründen Sie Ihre Antworten.

Aufgabe 2:

a) Die loop-Anweisung kann in WHILE-Programmen durch while-Anweisungen ersetzt werden. Beweisen Sie diese Aussage für die folgende loop-Anweisung:
loop a do P enddo (P sei ein beliebiges WHILE-Programm.)

b) Terminieren GOTO-Programme immer? Begründen Sie Ihre Antwort.

c) Geben Sie eine Turingmaschine $M = (Z, \{|, 0\}, \Gamma, \delta, z_0)$ mit einem Band an, die überprüft, ob die Anzahl der 0-Symbole im Eingabewort gerade ist.

Aufgabe 3:

Die Multiplikationsfunktion für zwei ganze Zahlen n, m kann durch fortgesetzte Addition realisiert werden:

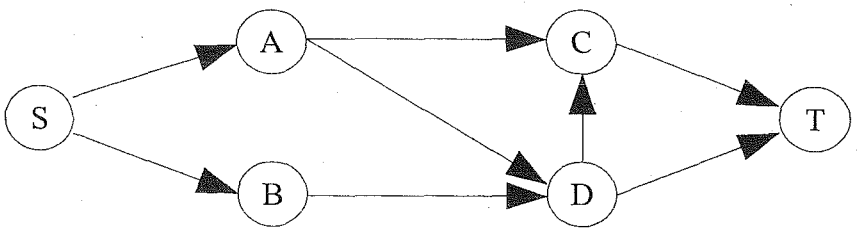
$$n \cdot m = mult(n, m) = \begin{cases} m + m \cdot (n-1) : falls\ n > 0 \\ 0 : sonst \end{cases}$$

Geben Sie eine **iterative** und eine **rekursive** Implementierung der Multiplikationsfunktion in einer imperativen oder objektorientierten Programmiersprache an. Verwenden Sie dabei nicht den Multiplikationsoperator dieser Programmiersprache.

Fortsetzung nächste Seite!

Aufgabe 4:

Repräsentieren Sie den folgenden Graphen mit Hilfe einer Adjazenzmatrix und einer Adjazenzliste.



Aufgabe 5:

Fügen Sie nacheinander die Schlüssel $7, 20, 30, 2, 14, 18, 9, 12, 4, 6, 16$ in einen anfangs leeren binären Suchbaum ein. Zeichnen Sie den kompletten Baum.

Aufgabe 6:

Gegeben seien die Zahlen $2, 3, 8, 5, 6, 4, 1, 7$. Sortieren Sie diese in aufsteigender Reihenfolge mit dem Sortierverfahren *Mergesort*. Geben Sie alle Zwischenschritte in einer Tabelle an, in der Sie die durch Aufteilung entstehenden Teilfolgen durch senkrechte Striche trennen.

Aufgabe 7:

Implementieren Sie die angegebenen Methoden einer Klasse Queue für Warteschlangen. Eine Warteschlange soll eine unbeschränkte Anzahl von Elementen aufnehmen können. Elemente sollen am Ende der Warteschlange angefügt und am Anfang aus ihr entfernt werden.

Sie können davon ausgehen, dass eine Klasse QueueElement mit der folgenden Schnittstelle bereits implementiert ist.

```
class QueueElement {
     QueueElement(Object contents);
     Object getContents();
     QueueElement getNext();
     void setNext(QueueElemente next);
}
```

Von der Klasse Queue ist folgendes gegeben:

```
class Queue {
     QueueElement first;
     QueueElement last;
}
```

Fortsetzung nächste Seite!

a) Schreiben Sie eine Methode `void append (Object contents)`, die ein neues Objekt in die Warteschlange einfügt.

b) Schreiben Sie eine Methode `Object remove ()`, die ein Element aus der Warteschlange entfernt und dessen Inhalt zurückliefert. Berücksichtigen Sie, dass die Warteschlange leer sein könnte.

c) Schreiben Sie eine Methode `boolean isEmpty ()`, die überprüft, ob die Warteschlange leer ist.

Aufgabe 8:

Gegeben sei folgende Funktion zur Berechnung der Quadratwurzel einer positiven, rellen Zahl $a \geq 1$: Es lässt sich zeigen, dass bei Terminierung der `while`-Schleife $x = y$ gilt. Diese Tatsache

```
double heron(double a) {
     double x, y;
     x = a; y = 1;
     while (x>y) {
          x = (x+y)/2;
          y = a/x;
     }
     return x;
}
```

können Sie für Ihre Lösung ausnutzen.

a) Bestimmen Sie eine Vorbedingung, eine Nachbedingung und die Invariante der `while`-Schleife.

b) Begründen Sie, warum aus der Invariante und der Nachbedingung folgt, dass der Algorithmus tatsächlich die Quadratwurzel von a berechnet.

Thema Nr. 2

Aufgabe 1:

a) Beschreiben Sie in Pseudocode oder einer Programmiersprache Ihrer Wahl einen *Greedy*-Algorithmus, der einen Betrag von n Cents mit möglichst wenigen Cent-Münzen herausgibt. Bei $n = 29$ wäre die erwartete Antwort etwa $1 \times 20ct, 1 \times 5ct, 2 \times 2ct$.

b) Beweisen Sie die Korrektheit Ihres Verfahrens, also dass tatsächlich die Anzahl der Münzen minimiert wird.

c) Nehmen wir an, Bayern führe eine Sondermünze im Wert von $7ct$ ein. Dann liefert der naheliegende Greedy-Algorithmus nicht immer die minimale Zahl von Münzen. Geben Sie für dieses Phänomen ein konkretes Beispiel an und führen Sie aus, warum Ihr Beweis aus Aufgabenteil a) in dieser Situation nicht funktioniert.

Aufgabe 2:

Sei $\sum$ endliches Alphabet. Sei $L \subseteq \sum^*$ eine reguläre Sprache über $\sum$. Die Sprache L^{rev} besteht aus allen Wörtern der Form $a_1...a_n$, wobei $a_n...a_1 \in L$ ist, also die Menge aller Wörter die, wenn von rechts nach links gelesen, in L sind.

Die Sprache L^{suf} bezeichnet alle Wörter w, deren sämtliche Suffixe in L sind; für die also gilt: wenn immer $w = uv$, so folgt $v \in L$.
Zeigen Sie, dass auch L^{rev} und L^{suf} regulär sind.

Aufgabe 3:

Sei $\sum = \{0, 1, 2, ..., t\}$ und $L \subseteq \sum^*$ beliebige kontextfreie Sprache. Sei L^{ord} die Menge der absteigend geordneten Wörter aus L, das sind diejenigen Wörter $a_1...a_n \in L$ für die gilt $a_1 \geq a_2 \geq ... \geq a_n$ an. Zeigen Sie, dass L^{ord} kontextfrei ist.

Aufgabe 4:

Das Partyveranstaltungsproblem ist das folgende. Gegeben ist eine Menge B von Bekannten und eine Menge $H \subseteq B \times B$ von (Paaren von) Bekannten, die sich nicht leiden können. Es ist festzustellen, ob eine Auswahl $U \subseteq B$ von k Bekannten ($|U| = k$) existiert (die Gästeliste), sodass in dieser keine zwei Bekannten enthalten sind, die sich nicht leiden können. Man zeige, dass das Partyveranstaltungsproblem NP-vollständig ist.