Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: ____________	**Herbst**	
Kennwort: ____________		**66112**
	1999	
Arbeitsplatz-Nr.: ________		

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: **Informatik (vertieft studiert)**

Einzelprüfung: **Automatentheorie,Komplexität,Algorith.**

Anzahl der gestellten Themen (Aufgaben): 2

Anzahl der Druckseiten dieser Vorlage: 4

Bitte wenden!

Thema Nr. 1

Sämtliche Teilaufgaben sind zu bearbeiten!

1. Gegeben seien die Sprache $L = \{ a^{2n} b^{3n} \mid n > 0\}$ über $\Sigma = \{a, b\}$,
die Grammatiken G_i ($i = 1, 2$) mit Startvariablen S_i und Produktionenmengen
$P_1 = P_0 \cup \{A \text{->} aa, B \text{->} bbb\}$,
$P_2 = P_0 \cup \{S_2 \rightarrow LS_1R, LA \rightarrow aa, aA \rightarrow aaa, aB \rightarrow abbb, bB \rightarrow bbbb, bR \rightarrow b\}$,
wobei $P_0 = \{S_1 \rightarrow ABS_1, S_1 \rightarrow AB, BA \rightarrow AB\}$

   a. Beweisen Sie:
   a1) $L \subseteq L(G_1)$,
   a2) $L \subseteq L(G_2)$.

   b. Geben Sie ein Wort in $L(G_1)$ an, das nicht Element von $L(G_2)$ ist.

   c. Ist L regulär? (Begründung!)

   d. Ist L kontextfrei? (Begründung!)

   e. Ist L kontextsensitiv? (Begründung!)

2. Zeigen Sie:
Ist die Funktion $f : \mathbf{N}^2 \rightarrow \mathbf{N}$ (**N** Menge der natürlichen Zahlen einschließlich 0) LOOP-berechenbar (primitiv rekursiv), so ist auch die Funktion $g: \mathbf{N}^2 \rightarrow \mathbf{N}$ mit

$$g(x, y) = \prod_{i=0}^{y} f(x, i)$$

LOOP-berechenbar (primitiv-rekursiv).

3. Welche der folgenden Eigenschaften sind für (deterministische) Turingmaschinen M entscheidbar? (Begründung!)

   a. M terminiert bei Eingabe 1999.
   b. M berechnet eine LOOP-berechenbare Funktion.
   c. Zu der von M berechneten Funktion $f : \mathbf{N}^2 \rightarrow \mathbf{N}$ gibt es ein f berechnendes WHILE-Programm mit höchstens zwei (ineinander geschachtelten) WHILE-Schleifen.

4. Erklären und vergleichen Sie zwei Ihnen bekannte Parameter-Übergabe-Mechanismen bei Prozeduraufrufen.

Thema Nr. 2

Sämtliche Teilaufgaben sind zu bearbeiten!

Aufgabe 1:

a) Geben Sie einen Datentyp für einfach verkettete Listen von ganzen Zahlen an. Als Programmiersprache können Sie dafür Pascal, Modula, oder C wählen. Verwenden Sie zur Definition des Datentyps das Pointerkonzept der von Ihnen gewählten Sprache.

b) Schreiben Sie eine Prozedur oder eine Funktion `insert`, die eine ganze Zahl `i` in eine aufsteigend geordnete Liste `l` an die korrekte Stelle einordnet;
z.B. liefert `insert` von 3 in <1, 2, 2, 7, 8> die Liste <1, 2, 2, 3, 7, 8>.

c) Schreiben Sie eine Prozedur oder Funktion `löschen`, die in einer Liste `l` eine ganze Zahl `i` an der Stelle löscht, an der `i` in `l` zuerst vorkommt.

Aufgabe 2:

a) Erklären Sie die Begriffe von Zeit- und Speicherplatz-Komplexität.

b) Ein bekanntes Spiel trägt den Namen "Türme von Hanoi". Das Spiel wird mit n Scheiben verschiedener Größe gespielt, die auf die Stäbe A, B, C gesteckt werden können. Zu Beginn stecken alle Scheiben auf Stab A und zwar derart, dass jeweils eine kleinere Scheibe auf einer größeren Scheibe liegt. Die Aufgabe besteht darin, die Scheiben von Stab A nach Stab B umzustecken, wobei folgende zwei Regeln zu beachten sind:

(1) In jedem Schritt darf nur genau eine Scheibe bewegt werden,

(2) eine größere Scheibe darf nie auf einer kleineren Scheibe liegen.

Der Algorithmus, der die Zugfolge beschreibt, ist wie folgt:

```
hanoi(int n, var Stab quelle, var Stab ziel, var Stab via);
    if n = 1 then
        /*bewege Scheibe von Quelle nach Ziel*/;
    else begin
        hanoi(n-1, quelle, via, ziel);
        /*bewege Scheibe von Quelle nach Ziel*/;
        hanoi(n-1, via, ziel, quelle);
    end;
end hanoi
```

Beweisen Sie, dass die Zeitkomplexität von `hanoi` exponentiell ist. Sie können annehmen, dass das Bewegen von Scheiben immer 1 Zeiteinheit braucht.
Hinweis: Geben Sie eine untere und eine obere Schranke für die Zeitkomplexität von `hanoi` an und verwenden Sie vollständige Induktion zum Beweis der Schranken.

Fortsetzung nächste Seite!

Aufgabe 3:

a) Sei $\{l_n\}_{n \in \mathbb{N}}$ eine Familie von Sprachen, definiert durch

$l_n = \{w \in \{0, 1\}^* \mid |w| \geq n \geq 1$ und das n-letzte Zeichen von w ist $1\}$.

Dabei bezeichnet $|w|$ die Länge des Wortes w.

a1) Geben Sie die allgemeine Form eines nichtdeterministischen endlichen Automaten für l_n an.

a2) Geben Sie einen deterministischen endlichen Automaten für l_3 an.

a3) Geben Sie den regulären Ausdruck r_3 für l_3 an (d.h. $L(r_3) = l_3$).

b) Gegeben sei die Sprache $L = \{w \in \{a,b\}^* \mid |w|_b = 2 \cdot |w|_a\}$, d.h. w ist ein Wort von L, wenn die Anzahl der b's in w gleich zweimal die Anzahl der a's in w ist.

b1) Geben Sie eine kontextfreie Grammatik Γ an, die die Sprache L erzeugt, d.h. $L = L(\Gamma)$. (Hinweis: Die Regel $S \rightarrow \varepsilon$, wobei S das Startsymbol bezeichnet, ist zugelassen.) Begründen Sie, warum Ihre Grammatik die Sprache L erzeugt.

b2) Wandeln Sie diese Grammatik in eine Backus-Naur-Form um. (Dabei ist sowohl die BNF- wie EBNF- Schreibweise zulässig.)

b3) Geben Sie die Ableitung des Wortes w = babbab an.

Aufgabe 4:

a) Erläutern Sie die auf Floyd und Hoare zurückgehende Verifikationsmethode. (In Ihrer Antwort müssen Sie mindestens die Begriffe von Zusicherung, schwächste Vorbedingung und Prädikattransformation erklären.)

b) Betrachten Sie folgendes Programmfragment P mit der Vorbedingung $V \equiv (x = a \;\&\; y = b)$ und der Nachbedingung $N \equiv (z = a \cdot b)$, wobei a, b ganze Zahlen sind.

```
z := 0;
while x <> 0 do
    begin z := z + y; x := x - 1 end;
```

b1) Zeigen Sie, dass die Formel $I \equiv (x \cdot y + z = a \cdot b)$ eine Schleifeninvariante ist.

b2) Beweisen Sie die partielle Korrektheit von P bezüglich der Vorbedingung V und der Nachbedingung N.

b3) Was fehlt noch für eine vollständige Verifikation? Terminiert dieses Programm immer? Begründen Sie Ihre Antwort.