| Prüfungsteilnehmer | Prüfungstermin | Einzelprüfungsnummer |
|---|---|---|

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

**Herbst**

**2000**

**66112**

# 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: 5

Bitte wenden!

**Thema Nr. 1**

**Sämtliche Teilaufgaben sind zu bearbeiten!**

**Aufgabe 1:** (Lexikalischer Vergleich)

Listen von Listen ganzer Zahlen kann man lexikalisch ordnen. Eine Liste $[x_1, ..., x_m]$ heißt lexikalisch kleiner als eine Liste $[y_1, ..., y_n]$, falls es ein i mit $0 \leq i \leq \min(m, n)$ so gibt, dass gilt:

$x_k = y_k$ für alle $k = 1, ..., i$ und
entweder $i = m < n$ oder $x_{i+1} < y_{i+1}$.

Wählen Sie für die folgenden Teilaufgaben a) - d) eine beliebige (jedoch für alle Teilaufgaben dieselbe) Programmiersprache.

a) Geben Sie einen Datentyp für Listen von ganzen Zahlen und einen Datentyp für Listen von Listen von ganzen Zahlen an.

b) Programmieren Sie den lexikalischen Vergleich `lex` auf Listen von ganzen Zahlen als eine Funktion oder Prozedur.

c) Schreiben Sie eine Prozedur oder eine Funktion `lex_insert`, die eine Liste `l` von ganzen Zahlen in eine lexikalisch geordnete Liste `ll` von Listen von ganzen Zahlen an die korrekte Stelle einsortiert.
Z. B. liefert die Anwendung von `lex_insert` auf die Argumente [1, 3] und [[1, 2], [1, 7], [4]] die Liste [[1, 2], [1, 3], [1, 7], [4]] als Ergebnis.

d) Schreiben Sie eine Prozedur oder Funktion `insert_sort`, die eine Liste von Listen von Zahlen `ll` lexikalisch sortiert. Die Prozedur oder die Funktion soll `lex_insert` verwenden.

**Aufgabe 2:** (Berechenbarkeit)

Seien $A, B \subseteq N_0$ rekursiv aufzählbar. Beweisen Sie die folgenden Behauptungen:

a) $A \cup B$ ist rekursiv aufzählbar.

b) $A \times B$ ist rekursiv aufzählbar.

Fortsetzung nächste Seite!

**Aufgabe 3:** (Sprachen und Automaten)

a) Zeigen Sie, dass die Sprache

$L = \{a^m b^n$: (m ist durch 3 teilbar und n ist ungerade) oder (m ist gerade und $n = 0)\}$

regulär ist. Geben Sie den regulären Ausdruck an.

b) Konstruieren Sie einen endlichen nichtdeterministischen Automaten, der L erkennt.

c) Konstruieren Sie einen endlichen deterministischen Automaten, der L erkennt.

d) Sei $L_1 = \{a^m b^n$: m, $n > 0$ und $m/n = 2/3\}$. Ist $L_1$ vom Typ Chomsky-2, ist $L_1$ zugleich vom Typ Chomsky-3? Beweisen Sie Ihre Behauptung.

**Aufgabe 4:** (Programmierung)

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

b) Betrachten Sie folgendes Programmfragment mit der Vorbedingung $V \equiv (s = 0 \,\&\, i = 0)$ und der Nachbedingung $N \equiv (s = (n+1)*n/2)$, wobei i, n und s ganze Zahlen sind.

```
while i <=n do
  begin
     s: = s + i;
     i: = i + 1
  end.
```

b1) Zeigen Sie, dass $i \leq n+1 \,\&\, s = (i-1)*i/2$ eine Schleifeninvariante ist, und beweisen Sie, dass diese erhalten bleibt.

b2) Beweisen Sie die Nachbedingung.

b3) Terminiert das Programm immer? Beweisen Sie Ihre Antwort.

## Thema Nr. 2

**Sämtliche Teilaufgaben sind zu bearbeiten!**

1. Geben Sie Zustandsdiagramme für deterministische endliche Automaten für die folgenden Sprachen an.

   a) Die Menge der Wörter $w \in \{a, b\}^*$, deren Länge $|w|$ entweder durch 2 oder 3 ohne Rest teilbar ist.

   b) Sei $L$ die Sprache der Aufgabe 1.a); d. h.

   $$L = \{w \in \{a, b\}^* : |w| \text{ entweder teilbar durch 2 oder 3}\}$$

   Sei $\overline{L} = \{a, b\}^* - L$ das Komplement von $L$. Geben Sie einen regulären Ausdruck für die Sprachen $L$ und $\overline{L}$ an.

2. Beweisen Sie, dass die Sprache $\{a^{2m}b^m : m \in \mathrm{N}\}$ kontextfrei, aber nicht regulär ist.

3. Skizzieren Sie in Pseudocode die Implementierung einer Operation, die aus einer verketteten Liste die Elemente entfernt, die mehrfach vorkommen. Geben Sie die notwendigen Datenstrukturen an, wobei die verkettete Liste mit Zeigern (Pointers) implementiert werden soll.

4. In dieser Aufgabe werden in Pseudocode die Datenstruktur und Implementierung von bestimmten Operationen für binäre Bäume gegeben.

   a) Definieren Sie in Pseudocode die Datenstruktur für die Implementierung von binären Bäumen (`BTree`), wobei Blätter dadurch dargestellt werden, dass `left` und `right` den Wert `null` haben.

   b) Implementieren Sie eine Methode

   ```
   void printInorder()
   ```

   für `BTree`, die den Baum in Infixdarstellung ausgibt.

   c) Geben Sie den Code an, der die Variable `t` mit Datentyp `BTree` mit einem Baum belegt, dessen Infixdarstellung `2+3*4` ist.

Fortsetzung nächste Seite!

5. Geben Sie eine kontextfreie Grammatik (oder BNF) an für die Menge der Palindrome gerader Länge über $\{a, b, c\}$ wobei die Mitte durch . markiert ist, also z. B. ab. ba. Skizzieren Sie die Implementierung eines *recursive descent* Parsers in Pseudocode.

6. Diese Aufgabe bezieht sich auf das Sortierungsverfahren `quicksort`.

   a) Geben Sie Pseudocode für `quicksort` an.

   b) Simulieren Sie die Schritte von `quicksort` auf der Eingabe 4, 2, 1, 3, 8, 5, 0, 6, 7.

   c) Was ist das *worst-case*-Verhalten von `quicksort`?

   d) Für welche Eingaben zeigt `quicksort` das *worst-case*-Verhalten?

   e) Was ist das *average-case*-Verhalten von `quicksort`?