Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: ______________	**Frühjahr 2004**	**66112**
Kennwort: ______________		
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: 7

Bitte wenden!

Thema Nr. 1

Sämtliche Teilaufgaben sind zu bearbeiten!

Aufgabe 1 (Automatentheorie und formale Sprachen, 30 Punkte)
Gegeben sei ein nichtdeterministischer endlicher Automat

$$N_1 = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b\}, \delta, q_0, \{q_2\})$$

wobei δ durch folgende Tabelle definiert ist. Z. B. geht der Automat aus dem Zustand q_0 (Auswahl der Spalte) durch Lesen eines „a" (Auswahl der Zeile) in die Zustände q_0 oder q_3 über:

δ	q_0	q_1	q_2	q_3	q_4	q_5
a	$\{q_0, q_3\}$	$\{q_2\}$	$\{q_2\}$	$\{q_0\}$	$\{q_4\}$	$\{q_5\}$
b	$\{q_1\}$	$\{q_5\}$	$\{q_5\}$	$\{q_1\}$	$\{q_1\}$	$\{q_5\}$
ε	$\{q_4\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

a) Zeichnen Sie den gegebenen nichtdeterministischen endlichen Automaten.

b) Wandeln Sie den nichtdeterministischen endlichen Automaten N_1 durch Anwendung der Teilmengenkonstruktion in einen deterministischen endlichen Automaten D_1 um. Zeichnen Sie D_1.

c) Wenden Sie den Table-Filling-Algorithmus oder ein anderes Verfahren zur Minimalisierung auf D_1 an und fassen Sie alle äquivalenten Zustände zusammen. Zeichnen Sie den resultierenden Automaten.

d) Welche Sprache erkennen die Automaten? Geben Sie einen möglichst kurzen regulären Ausdruck zur Beschreibung dieser Sprache an!

Aufgabe 2 (Automatentheorie und formale Sprachen, 30 Punkte)

Wir betrachten die Sprache

$$L = \{a^m b^n ;\ n, m \in \mathbb{N}_0,\ m<n\}$$

a) Beweisen Sie, dass diese Sprache nicht regulär ist.
Geben Sie eine kontextfreie Grammatik an, die L erzeugt.

b) Geben Sie einen Kellerautomaten an, der die Sprache erkennt, und zeichnen Sie sein Übergangsdiagramm.

Fortsetzung nächste Seite!

Aufgabe 3 (Berechenbarkeit, 30 Punkte)

a) Erläutern Sie informell wie man mit Hilfe einer 1-Band-Turingmaschine eine k-Band-Turingmaschine simulieren kann.

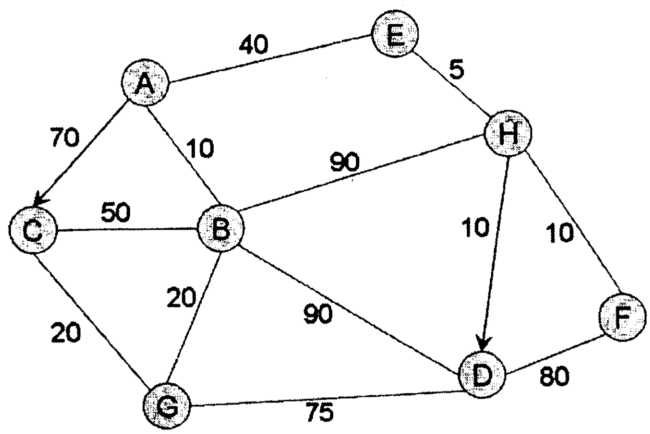
b) Begründen Sie, warum WHILE-Programme mächtiger als LOOP-Programme sind. (Die Begriffe LOOP-Programm und FOR-Programm werden in der Literatur synonym gebraucht.)

c) Erläutern Sie, wie mit Hilfe des Reduktionsprinzips die Unentscheidbarkeit eines Problems gezeigt werden kann.

Aufgabe 4 (Algorithmen, 30 Punkte)

Ein wichtiges Problem im Bereich der Graphalgorithmen ist die Berechnung kürzester Wege.

Gegeben sei der folgende Graph, in dem Städte durch Kanten verbunden sind. Die Kantengewichte geben Fahrzeiten an. Außer den durch Pfeile als nur in eine Richtung befahrbar gekennzeichneten Straßen sind alle Straßen in beiden Richtungen befahrbar.



a) Geben Sie zu dem obigen Graphen zunächst eine Darstellung als Adjazenzmatrix an.

b) Berechnen Sie nun mit Hilfe des Algorithmus von Dijkstra die kürzesten Wege vom Knoten A zu allen anderen Knoten.

Aufgabe 5 (Datenstrukturen, 30 Punkte)

Hashtabellen bilden eine effiziente Datenstruktur für das so genannte Wörterbuchproblem, bei dem auf einer Menge von Objekten die Operationen *insert, delete* und *member* benötigt werden.

a) Skizzieren Sie die drei üblichen Vorschläge zur Behandlung von Kollisionen beim geschlossenen Hashing. Welche Probleme bringen die einzelnen Vorschläge mit sich und weshalb entstehen manche Probleme bei den elaborierteren Vorschlägen nicht?

Fortsetzung nächste Seite!

b) Beschreiben Sie, wie Löschoperationen in Hashtabellen realisiert werden können. Was ist hierbei zu beachten?

c) Beschreiben Sie eine Hashfunktion für den Fall, dass die Schlüssel der zu verwaltenden Objekte Zeichenketten sind (d. h. die Hashfunktion soll aus der Menge der Zeichenketten in die Menge der Zellenadressen abbilden). Begründen Sie die Wahl Ihrer Funktion.

d) Eine Alternative zu Hashverfahren bilden binäre Suchbäume. Diese können jedoch unausgeglichen werden. Deshalb wurden verschiedene Ansätze für balancierte Bäume entwickelt. Stellen Sie eine Art von balancierten Bäumen vor. Geben Sie hierzu die Struktureigenschaften der balancierten Bäume möglichst präzise an und skizzieren Sie knapp, wie diese Struktureigenschaften bei Einfügeoperationen erhalten bleiben.

Aufgabe 6 (Systementwurf, 30 Punkte)

a) Geben Sie exemplarisch drei Design-Pattern (oder Entwurfsmuster) an, erläutern Sie deren Einsatzgebiet und Funktionsweise und diskutieren Sie anhand dieser Beispiele verschiedene mögliche Arten von Design-Pattern.

b) Erstellen Sie ein Klassendiagramm (im Sinne der UML) für die Prüfungsverwaltung an einer Hochschule. Sie sollten dabei zumindest Studenten, Prüfer, mündliche Prüfungen, schriftliche Prüfungen und Prüfungsfächer modellieren. Nutzen Sie die Möglichkeiten der Vererbung und bemühen Sie sich um eine möglichst flexible Modellierung. Begründen Sie dabei einzelne Entwurfsentscheidungen.

Thema Nr. 2

Sämtliche Teilaufgaben sind zu bearbeiten!

Aufgabe 1: (Arraysumme)

a) Schreiben Sie eine Methode in Java oder einer anderen imperativen Programmiersprache, die als Parameter ein Integer-Array (beliebiger Länge) erwartet und als Wert die Summe der Array-Elemente zurück gibt. Es wird nicht nur die Korrektheit des Programms, sondern auch der Programmierstil bewertet!

b) Geben Sie die Schleifeninvariante an, die man für einen Korrektheitsbeweis des Programms im Hoare-Kalkül benutzen würde.

c) An welchen Stellen im Programm muss die Schleifeninvariante gelten?

Aufgabe 2: (Breitensuche in Binärbäumen)

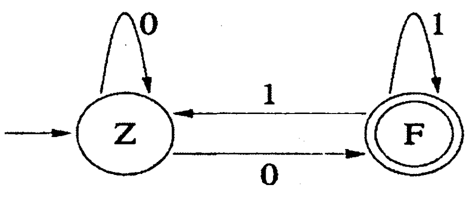
Benutzen Sie eine funktionale Programmiersprache Ihrer Wahl.

a) Definieren Sie einen Datentyp für einen Binärbaum, dessen Knoten mit Integerwerten behaftet sind.

b) Programmieren Sie eine Funktion, die einen solchen Binärbaum als Parameter erwartet und seine Knotenwerte in einer Liste in Breitenordnung zurück gibt. Es wird nicht nur die Korrektheit des Programms, sondern auch der Programmierstil bewertet!

Aufgabe 3: (Endliche Automaten)

Wir betrachten den folgenden endlichen Automaten:



a) Der Automat ist nichtdeterministisch. Es gibt ein Verfahren, mit dem man einen deterministischen aus einem nichtdeterministischen Automaten gewinnt.

   i) Beschreiben Sie das Prinzip dieses Verfahrens kurz.
   ii) Wenn der nichtdeterministische Automat n Zustände hat, wieviele Zustände kann der abgeleitete deterministische Automat im schlimmsten Fall haben?

b) Wenden Sie das Konstruktionsverfahren auf den obigen Automaten an. Minimieren Sie das Ergebnis, indem Sie alle unerreichbaren Zustände durchstreichen. Der deterministische Automat muss nur als Grafik angegeben werden.

c) Beschreiben Sie mit einem regulären Ausdruck die Sprache, die diese beiden Automaten erkennen.

Fortsetzung nächste Seite!

Aufgabe 4: (Abstiegsfunktion)

Hier ist eine Funktion (in Haskell kodiert), die angibt, ob die ihr übergebene ganze Zahl gerade oder ungerade ist:

```
even :: Int -> Bool
even x = if x<0 then even (-x)
                else if x==0 then True
                              else if x==1 then False
                                           else even (x-2)
```

Geben Sie eine Abstiegsfunktion *h* an, die die Terminierung von even für alle Eingabewerte belegt:

a) Geben Sie die Funktionsvorschrift von *h* an.
b) Zeigen Sie, dass *h* die Eigenschaften einer Abstiegsfunktion erfüllt.

Aufgabe 5: (Sortieren)

In dieser Frage betrachten wir Verfahren zum Sortieren eines Arrays von Werten aus einer total geordneten Menge.

a) Wie ist präzise definiert, dass eine Funktion g in $O(f(n))$ ist?

b) Beim Heapsort-Verfahren bringt eine Funktion *heapify* das Eingabe-Array in eine bestimmte Form. Beschreiben Sie diese "Heap-Eigenschaft" kurz und präzise.

c) Zu welcher O-Klasse gehören die folgenden beiden Sortierverfahren?
Bitte eine kurze Begründung angeben.

   i) Quicksort
   ii) Heapsort

Aufgabe 6: (Formale Sprachen)

Wir betrachten die folgende Sprache symmetrischer Zeichenfolgen über dem Alphabet $\{0, 1\}$:

$$\left\{x_1 \ldots x_n x_n \ldots x_1 \mid n > 0 \wedge \left(\forall i : 0 < i \leq n : x_i \in \{0,1\}\right)\right\}$$

a) Zu welcher Typklasse (Typ 0-3) gehört diese Sprache?

b) Wann ist eine Grammatik eindeutig?

c) Geben Sie eine eindeutige Grammatik für die Sprache an.

d) Mit welcher Art von Automaten lässt sich diese Sprache erkennen?
Grenzen Sie die Automatenart so präzise wie möglich ein.

Fortsetzung nächste Seite!

Aufgabe 7: (NP-Vollständigkeit)

a) Was bedeutet es für ein Problem, NP-hart zu sein?

b) Definieren Sie präzise den Begriff der NP-Vollständigkeit.

c) Begründen Sie den folgenden Satz:

$$(A \text{ NP-vollständig} \wedge A \in P) \iff P = NP$$