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

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

**Frühjahr 2008**

**46114**

# Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

## — Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Algorithmen/Datenstrukturen/Programmiermethoden**

Anzahl der gestellten Themen (Aufgaben): **2**

Anzahl der Druckseiten dieser Vorlage: **10**

**Bitte wenden!**

## Thema Nr. 1

### Aufgabe 1: (Rekursion und Komplexität)

In einer funktionalen Programmiersprache Ihrer Wahl:

a) Definieren Sie rekursiv die Funktion *app*, die zwei Listen verbindet!
Beispiel: *app* ([1,2,3],[4,5,6]) = [1,2,3,4,5,6]

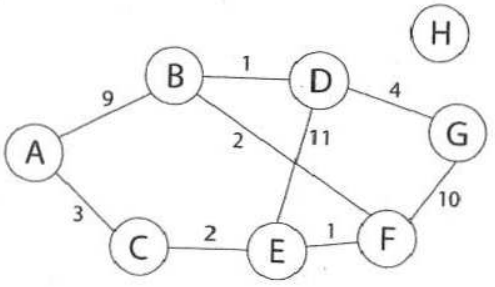
b) Welche Zeitkomplexität im schlimmsten Fall hat *app*? Geben Sie die Aufwandsklasse in der O-Notation in Abhängigkeit von der Größe der beiden Eingabeparameter an und begründen Sie Ihre Aussage!

c) Definieren Sie auch rekursiv die Funktion *rev*, die eine Liste revertiert!
Beispiel: *rev* [1,2,3,4] = [4,3,2,1]

d) Welche Zeitkomplexität im schlimmsten Fall hat *rev* in Abhängigkeit von dem Eingabeparameter (mit Begründung)?

### Aufgabe 2: (Dijkstra-Algorithmus für kürzeste Wege)

Führen Sie auf folgendem Graphen den Algorithmus von Dijkstra zur Bestimmung der kürzesten Wege aus! Der Startknoten ist *A*. Geben Sie dabei nach jedem Schritt den Zustand des Heaps an! Geben Sie am Ende den kürzesten Pfad von *A* zu jedem Knoten sowie dessen Gewicht an!




### Aufgabe 3: (Java-Arrays und Verifikation)

a) Implementieren Sie in Java eine Methode

```
int maximumPosition(int[] a),
```

die den kleinsten Index (Position) im Array `a` liefert, an der ein Element mit dem größten Wert gespeichert ist! Falls das Array leer ist, soll die Zahl `-1` zurückgegeben werden.
Beispiel: `maximumPosition`, angewendet auf das Array `{3,9,5,9}`, soll den Wert `1` liefern.

b) Wie lautet die Schleifeninvariante und an welchen Stellen des Programms muss sie gelten?

**Fortsetzung nächste Seite!**

## Aufgabe 4: (Abstiegsfunktion)

Hier ist eine rekursive 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 Signatur von *h* an!
b) Geben Sie die Funktionsvorschrift von *h* an!
c) Zeigen Sie, dass *h* die Eigenschaften einer Abstiegsfunktion erfüllt!

## Aufgabe 5: (Parameterübergabe)

Gegeben ist folgendes imperative Programm in Java-Syntax:

```
int   i;
int[] a;

void p(int k) {
   System.out.print(k);              // Pos 1
   i = 1;
   k = k+1;
   System.out.print(a[0]);           // Pos 2
   System.out.print(k);              // Pos 3
}

void main() {
   a = new int[] {2, 9};
   i = 0;
   p(a[i]);
   System.out.print(a[0]);           // Pos 4
   System.out.print(a[1]);           // Pos 5
}
```

Übertragen Sie die folgende Tabelle und geben Sie darin die Zahlen an, die das Programm an den kommentierten Programmpositionen unter den verschiedenen Parameterübergabearten ausgibt!

| call by | value | value-result | reference | name |
|---|---|---|---|---|
| Pos 1 | | | | |
| Pos 2 | | | | |
| Pos 3 | | | | |
| Pos 4 | | | | |
| Pos 5 | | | | |

**Fortsetzung nächste Seite!**

## Aufgabe 6: (Abstrakte Datentypen)

Betrachten wir einen axiomatisch definierten abstrakten Datentyp! Die *Signatur* vereinbart u. a. die Definitions- und Wertbereiche der Operationen, die für den Datentyp definiert sind. Die Bedeutungen (Semantik) der Operationen werden dann mit Gleichungen angegeben, die die Operationen zueinander in Beziehung setzen.

a) Geben Sie die Signaturen und die beschreibenden Gleichungen für die Operationen eines Kellers (Stacks) an, als da sind:
   - `push`: lege ein neues Element auf den Stack
   - `pop`: entferne ein Element vom Stack
   - `top`: gib den Wert des obersten Elements auf dem Stack an
   - `isempty`: gib an, ob der Stack leer ist oder nicht

b) Die Implementierung eines abstrakten Datentyps wird in der Regel vor dem Benutzer versteckt (Geheimnisprinzip). Geben Sie zwei Gründe dafür an.

## Aufgabe 7: (Heap und Suchbaum)

a) Gegeben ist ein MinHeap mit der Belegung

| 1 | 3 | 2 | 4 | 5 | 6 | 8 | 7 |
|---|---|---|---|---|---|---|---|

   Das Element 1 soll entfernt und danach die Heapeigenschaft wiederhergestellt werden. Geben Sie die Belegung des Heaps nach jeder einzelnen Vertauschung an!

b) Wir betrachten Eingabe und Löschen in einem binären Suchbaum.
   - Zeichnen Sie den Baum, der durch die folgende Eingabesequenz entsteht:

```
insert(Nadine); insert(Roland); insert(Don);
insert(Alfons); insert(Leo); insert(Petra);
insert(Dieter); insert(Martin); insert(Fredo);
insert(Gregor); insert(Berta); insert(Andrea);
insert(Chris); insert(Gerd);
```

   - Zeichnen Sie nun den Baum, der nach Löschen des Eintrags `Don` entsteht!

# Thema Nr. 2

## Aufgabe 1

Betrachten Sie die beiden rekursiven Algorithmen:

```
1 long fakultaet(int n){
            return (n==1) ? 1 : n * fakultaet(n-1);
      }
```

```
2 void lege(int scheibe, char quelle, char ziel, char ablage) {
      if (scheibe > 0){
            lege(scheibe - 1, quelle, ablage, ziel);
            System.out.println("Ich lege eine Scheibe von "
                  + quelle + " nach " + ziel);
            lege(scheibe - 1, ablage, ziel, quelle);
      }
  }
```

a) Welche Rekursionsform liegt jeweils vor?
b) Geben Sie den Algorithmus 1 in iterativer Form an!
c) Geben Sie für beide rekursiven Algorithmen eine Aufwandsabschätzung in O-Notation!

## Aufgabe 2

Gegeben ist folgendes JAVA-Programm, das den Gesamtlohn der Mitarbeiter einer Gärtnerei ermittelt:

```
1   abstract class Mitarbeiter {
2         private String name;
3         *    private double stdlohn = 11.50;
4         float std;
5         *    public void Mitarbeiter(String name){
6               this.name = name;
7         }
8
9         abstract double getLohn();
10
11        public void setStdLohn(double arg) {
12              this.stdlohn = arg;
13        }
14
15        public void setStd(float arg) {
16              this.std = arg;
17        }
18  }
19  public class MitList {
20        private static Mitarbeiter[] alle = new Mitarbeiter[3];
```

**Fortsetzung nächste Seite!**

```
21
22      static void set(Mitarbeiter m, int n){
23          alle[n] = m;
24      }
25
26      static double getGesamtlohn(){
27          double l = 0;
28          for (int i = 0; i < alle.length; i++) {
29              l += alle[i].getLohn();
30          }
31          return l;
32      }
33  }
34  public class Gaertner extends Mitarbeiter {
35      float ueberstd;
36
37      public double getLohn(){
38          return (this.std + this.ueberstd) * this.stdlohn;
39      }
40
41      public void setUeberstd(float x){
42          this.ueberstd = x;
43      }
44  }
45  public class Verkaeufer extends Mitarbeiter {
46
47      public double getLohn(){
48          return this.std * this.stdlohn;
49      }
50  }
51  public class Gaertnerei {
52
53      public static void main(String[] args) {
54          *    Gaertner albert = new Mitarbeiter("Albert");
55          *    Gaertner.setUeberstd(2);
56          Verkaeufer betti = new Verkaeufer("Betti");
57          betti.setStdLohn(12.50);
58          albert.setStd(38.5);
59          betti.setStd(42);
60
61          MitList.set(albert,0);
62          MitList.set(betti,1);
63          MitList.set(new Verkaeufer("Charlie"),2);
64
65          System.out.println(MitList.getGesamtlohn());
66      }
67
68  }
```

**Fortsetzung nächste Seite!**

a) Das gegebene Java-Programm enthält vier fehlerhafte Codezeilen, die mit einem Stern (*) markiert sind. Geben Sie die Zeilen korrekt an.

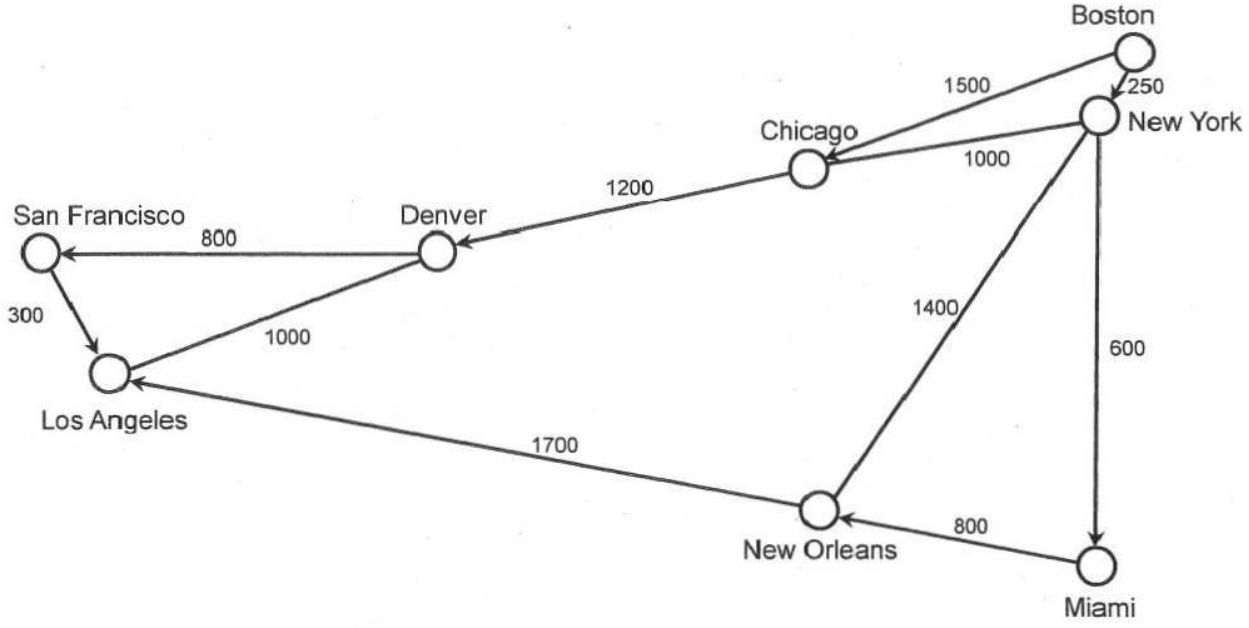
b) In den Klassen `Verkaeufer` und `Gaertner` fehlen jeweils die Konstruktoren. Geben Sie geeignete Konstruktoren an!

c) Erklären Sie das Konzept der polymorphen Operationen am Beispiel der Operation `getLohn()` in höchstens drei Sätzen!

d) An welchen Stellen des Programms wird Polymorphie von Variablen benutzt? Begründen Sie Ihre Antworten in höchstens drei Sätzen!

## Aufgabe 3

Gegeben ist folgende Darstellung von Flugentfernungen verschiedener Orte:




a) Stellen Sie die Adjazenzmatrix für den obigen Graphen auf! Beachten Sie insbesondere, dass der Graph sowohl gerichtete als auch ungerichtete Teilabschnitte enthält!

b) Erläutern Sie in höchstens drei Sätzen, welche Bedeutung eine vollständig mit nicht-negativen Zahlen gefüllte Zeile in der Adjazenzmatrix besitzt!

c) Stellen Sie den Graphen auch als Adjazenzliste dar!

d) Stellen Sie tabellarisch Vor- und Nachteile der Speicherung des Graphen unter Verwendung dieser beiden Darstellungsvarianten gegenüber!

e) Berechnen Sie nun mit Hilfe des Algorithmus von Dijkstra die kürzesten Wege von Boston zu allen anderen angegebenen Städten! Geben Sie alle Zwischenschritte an!

**Fortsetzung nächste Seite!**

f) Es ergeben sich während des Ablaufs des Algorithmus verschiedene Wege mit gleichen Entfernungen für einen bestimmten Zielort! Entscheiden Sie sich in diesem Fall systematisch für eine der Varianten und begründen Sie Ihre Entscheidung!

g) Begründen und erläutern Sie in höchstens drei Sätzen die Aussage: *„Bei der Implementierung bestimmter Teile des Algorithmus von Dijkstra kann ein Heap (eine Halde) vorteilhaft eingesetzt werden."*!

## Aufgabe 4

Im Folgenden werden drei rekursive Definitionen eines Binärbaumes angegeben. Nur die Definition I stellt eine korrekte Beschreibung eines Binärbaumes dar. Begründen Sie, warum die Definitionen II und III keine korrekten Spezifikationen sind. Illustrieren Sie Ihre Begründung jeweils mit einem Beispiel.

I. Ein Binärbaum ist ein Blatt oder ein Knoten mit zwei Bäumen als linkem und rechtem Sohn.
II. Ein Binärbaum ist ein Knoten mit zwei Bäumen als linkem und rechtem Sohn.
III. Ein Binärbaum besteht aus einem Wurzelknoten und endlich vielen weiteren Knoten. Jeder Knoten ist entweder ein Blatt oder wieder Wurzelknoten eines Baumes.

## Aufgabe 5

Ein Suchbaum sei definiert als Binärbaum mit folgender charakterisierender Eigenschaft:

„In einem binären Suchbaum gilt für jeden Knoten: Alle Schlüssel, die kleiner sind als der Schlüssel des Knotens werden im linken Teilbaum gespeichert, alle Schlüssel, die größer sind, im rechten Teilbaum. Ein Schlüssel kommt höchstens einmal vor."

Der abstrakte Datentyp (ADT) `TREE` enthalte mindestens die Signaturen:

```
create:                          -> TREE
bintree:  TREE x KEY x TREE      -> TREE
insert:   TREE x KEY             -> TREE
delete:   TREE x KEY             -> TREE
empty:    TREE                   -> Boolean
search:   TREE x KEY             -> Boolean
```

und mindestens die Axiome:

```
i,k: KEY, l,r,t: TREE
empty(create) = TRUE
insert(create,k) = bintree(create,k,create)
search(delete(t,k),k) = FALSE
```

a) Nennen Sie die wichtigsten Gründe für die Verwendung abstrakter Datentypen!

b) Ergänzen Sie das unvollständige Axiomensystem des ADT `TREE` für die Operation `insert`, so dass das Einfügen eines Schlüssels in einen binären Suchbaum gemäß der obigen Charakterisierung korrekt spezifiziert wird!

c) Fügen Sie die Schlüssel 5, 4, 2, 3, 8, 6, 5, 9, 1, 7 in einen anfangs leeren Suchbaum ein!

**Fortsetzung nächste Seite!**

Für die Operation dunno seien die Signatur und das folgende Axiom gegeben.

```
dunno: TREE -> KEY
k: KEY, l,r: TREE
dunno(bintree(l,k,r)) = { k          falls empty(r)
                        { dunno(r)   sonst
```

d) Beschreiben Sie kurz die Wirkungsweise und das Resultat der Operation dunno!

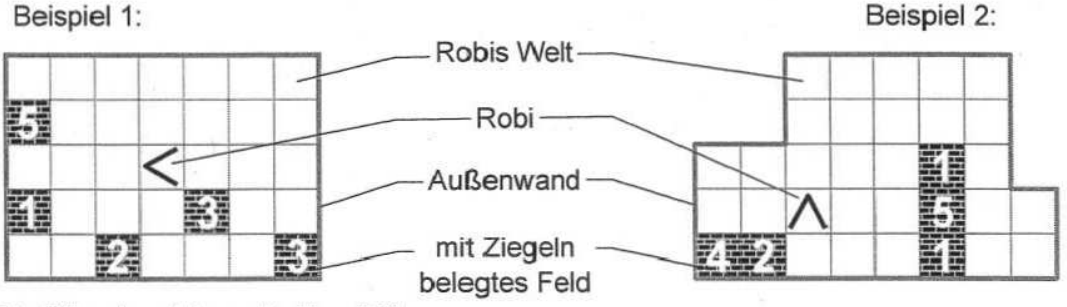
e) Schätzen Sie den Aufwand der Operation dunno im schlechtesten Fall (worst case) ab!

f) Wie kann der Suchbaum aufgebaut werden, um den worst case-Aufwand der Prozedur dunno und beliebiger Suchvorgänge zu verbessern? Skizzieren Sie eine gängige Vorgehensweise in wenigen Sätzen!

## Aufgabe 6

Zu Vermittlung der Grundlagen der Algorithmik im Unterricht kommen oft schulspezifische Lern- und Programmierumgebungen (wie z. B. Karol, der Roboter oder Kara, der Marienkäfer) zum Einsatz. Im Rahmen dieser Aufgabe sollen Sie eine solche Umgebung in Teilen entwerfen und implementieren. Die hier zu entwerfende Umgebung erhält den Namen *Robi, der Roboter*.

*Robi* ist ein virtuelles Robotersystem. Der Roboter *Robi* (dargestellt durch einen Pfeil „>“) befindet sich in einer, durch ausschließlich parallel zu der x- und y-Achse verlaufende Außenwände begrenzten, zusammenhängenden Welt. Die Pfeilspitze zeigt die momentane Blickrichtung an.




Die Ziffern bezeichnen die Ziegelhöhen.

Auf einem Feld der Welt können bis zu einer maximalen Höhe von *hmax* Ziegel abgelegt werden, die *Robi* in unbegrenzter Anzahl mit sich herumtragen kann. Robi kann einen Höhenunterschied von einem Ziegel überbrücken, d. h. er kann z. B. von einem leeren Feld ein mit einem Ziegel belegtes Feld betreten.

**Fortsetzung nächste Seite!**

*Robi* beherrscht folgende Funktionen:

- *geheVor* (*Robi* geht in seiner Welt in Blickrichtung einen Schritt vor, sofern dies möglich ist)
- *dreheLinks* (*Robi* dreht sich in seiner Welt auf aktueller Position um 90 Grad nach links)
- *nimmAuf* (*Robi* nimmt in seiner Welt einen Ziegel von dem Feld, das sich in Blickrichtung vor ihm befindet, sofern dies möglich ist)
- *legeAb* (*Robi* legt in seiner Welt einen Ziegel auf das Feld, das sich in Blickrichtung vor ihm befindet, sofern dies möglich ist)

und verfügt über folgende Sensoren:

- *vorneWand* (liefert true, wenn sich vor dem nächsten Feld in seiner Welt in Blickrichtung eine Außenwand befindet)
- *ziegelVorhanden* (liefert true, wenn sich auf dem nächsten Feld in seiner Welt in Blickrichtung mindestens ein Ziegel befindet)

a) Entwerfen Sie eine geeignete Datenstruktur zur Repräsentation von *Robis Welt*. Erläutern Sie diese kurz!

b) Geben Sie ein Klassendiagramm an, das den Zusammenhang zwischen *Roboter* und *Welt* mit allen problemrelevanten Attributen, Methoden und Kardinalitäten enthält! Erläutern Sie dieses kurz!

c) Geben Sie eine kommentierte Implementierung Ihres Modells an: Implementieren Sie die Klasse *Roboter* sowie alle dazu erforderlichen Bestandteile der Klasse *Welt*! Die Bildschirmdarstellung brauchen Sie nicht zu berücksichtigen. Verwenden Sie eine gängige (objektorientierte) Programmiersprache oder einen geeigneten Pseudocode.