Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____ Kennwort: _____ Arbeitsplatz-Nr.: _____	**Frühjahr** **2006**	**46114**

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Algorithmen/Datenstrukt./Progr.-meth.**

Anzahl der gestellten Themen (Aufgaben): 2

Anzahl der Druckseiten dieser Vorlage: 10

Bitte wenden!

Thema Nr. 1

Vorbemerkung:

Es wird empfohlen, zur Lösung der Programmieraufgaben eine der Programmiersprachen Pascal oder Java zu verwenden. Dazu ähnliche Programmiersprachen wie C oder C++ sind jedoch auch zugelassen.

Teilaufgabe A (Dateiverwaltung)

Adressen und Telefonnummern sollen in nach Namen sortierten Binärbäumen abgelegt werden. Beim Auftreten mehrerer Einträge unter gleichem Familiennamen werden zur Unterscheidung zunächst die Vornamen und schließlich, falls diese ebenfalls übereinstimmen, die Geburtsdaten verwendet. Mit diesen drei Merkmalen wird dann aber Eindeutigkeit vorausgesetzt, so dass das Tripel (Familienname, Vorname, Geburtsdatum) als Zugriffsschlüssel (*Schlüssel*) verwendet werden kann.
Familienname und Vorname werden als *String* (bei Programmierung in Java) bzw. als *string* (bei Programmierung in Pascal), das Geburtsdatum wird als *ganze Zahl* im Format JJJJMMTT eingegeben.
Als Schnittstelle für die Programmieraufgaben steht Ihnen die Arithmetik der ganzen Zahlen (einschließlich der Vergleichsoperationen) und genau eine Vergleichsoperation *kl(String a, String b)* (in Java) bzw. *kl(string a, string b)* (in Pascal) zwischen zwei Zeichenreihen zur Verfügung, die jeweils den Wahrheitswert true genau dann liefert, wenn *a* lexikographisch vor *b* kommt. Sind *a* und *b* gleich oder kommt *b* vor *a*, ist das Resultat der Wahrheitswert *false*.

1. Geben Sie eine für diese Teilaufgabe A geeignete Datenstruktur an! Adresse und Telefonnummer sollen dabei als *String* (Java) bzw. *string* (Pascal) eingegeben werden. Da das Tripel (Familienname, Vorname, Geburtsdatum) als Zugriffsschlüssel für die Einträge im sortierten Binärbaum dient, soll dabei für diese *Schlüssel* eine eigene Klasse (Java) bzw. ein eigener Verbund (Pascal) eingeführt werden.

2. Programmieren Sie unter dem Namen *eintrag* eine Instanzmethode (Java) bzw. eine Prozedur (Pascal), die folgender Spezifikation genügt:

   2.1 Eingabeparameter sind (zum *Schlüssel* zusammengefasst) Familienname, Vorname und Geburtsdatum einer Person sowie die zu dieser Person gehörende Adresse und Telefonnummer.
   Hinweis: Bei Programmierung in Pascal muss auch der entsprechende Binärbaum als Parameter übergeben werden.

   2.2 Die Methode bzw. die Prozedur trägt die Eingabedaten (nach 2.1) in den entsprechend 1. aufgebauten sortierten Binärbaum ein, falls es zu dem angegebenen *Schlüssel* noch keinen Eintrag gibt. Danach soll der Binärbaum wieder sortiert sein.
   Hinweis: Es ist empfehlenswert, für diese Programmieraufgabe die Schnittstelle zunächst um eine Methode bzw. eine Funktionsprozedur *kl1 (Schlüssel a, Schlüssel b)* zu erweitern.

Fortsetzung nächste Seite!

3. Programmieren Sie unter dem Namen *ausgabe* eine Instanzmethode (Java) bzw. eine Prozedur (Pascal), die folgender Spezifikation genügt:

   3.1 Eingabeparameter sind (zum *Schlüssel* zusammengefasst) Familienname, Vorname und Geburtsdatum einer Person.
   Beachten Sie den Hinweis bei 2.1!

   3.2 Die Methode bzw. die Prozedur druckt die zu den Daten nach 3.1 gehörende Adresse und Telefonnummer auf dem Bildschirm (Standard-Output) aus, falls zu der Eingabe ein Eintrag im Binärbaum enthalten ist. Andernfalls druckt sie die Meldung „`kein Eintrag`" aus.

4. Es soll eine Methode *loesche* entwickelt werden, die einen Eintrag in einem entsprechend 1. aufgebauten sortierten Binärbaum löscht.

   4.1 Programmieren Sie zunächst zu einem nichtleeren nach 1. aufgebauten sortierten Binärbaum *sb* eine Instanzmethode (Java) bzw. eine Funktionsprozedur (Pascal) unter dem Namen *max*, die folgender Spezifikation genügt:

      4.1.1 Bei Programmierung in Java hat diese Methode keinen Parameter; bei Programmierung in Pascal hat die Funktionsprozedur nur *sb* als Parameter.

      4.1.2 Die Methode bzw. die Funktionsprozedur liefert als Resultat den Teilbaum von *sb*, der in seiner Wurzel den maximalen Schlüssel von *sb* enthält.

   4.2 Formulieren Sie mit Hilfe der nunmehr gegebenen Schnittstelle (also einschließlich *max* nach 4.1), wie eine Methode bzw. eine Prozedur zu programmieren ist, die einen Eintrag zu einem gegebenen Schlüssel in einem sortierten Binärbaum löscht, so dass der Binärbaum danach wieder sortiert ist. Die Formulierung soll so konzis (d.h. knapp und unmissverständlich) sein, dass eine erfahrene Programmiererin (ein erfahrener Programmierer) das Programm ohne weiteres schreiben kann.

Fortsetzung nächste Seite!

Teilaufgabe B (Textverarbeitung)

Gegeben sei das Alphabet $I = (a, b, c, \ldots, z)$. Als Schnittstelle für Zeichenreihen aus I^* sowie für deren Bearbeitung stehen Ihnen einfach verkettete Listen mit den folgenden Methoden (Java) bzw. Funktionsprozeduren (Pascal) zur Verfügung:

In Java:

```
class Zr{
char c; Zr next;
static int laenge (Zr x)                          // liefert die Länge von x
static boolean istanfang (Zr x,Zr y){ . . . }     // liefert genau dann true,
                                                  // wenn y ein Anfang von x ist
static Zr copy (Zr x){ . . . }                    // liefert eine Kopie von x
}
```

bzw.

in Pascal:

```
type zr =↑ zrel;
type zrel =record c :char; next:zr end;
function laenge(x : zr) :integer;          (* liefert die Länge von x *)
function istanfang(x : zr,y : zr) :boolean; (* liefert genau dann true, *)
                                           (* wenn y ein Anfang von x ist *)
function copy(x : zr) : zr;                (* liefert eine Kopie von x *)
```

1. Programmieren Sie mit Hilfe der angegebenen Schnittstelle unter dem Namen *ersetze* eine Methode (Java) bzw. eine Prozedur (Pascal), die folgender Spezifikation genügt:

   1.1 Eingabeparameter sind drei nichtleere Wörter x, y und z aus I^*.

   1.2 Die Methode ersetzt das erste (von links gelesen) in x vorkommende y durch z und nennt das so entstehende Wort wieder x. Mit diesem neuen Wort x wird die Methode mit den (unveränderten) Wörtern y und z solange wiederholt, bis y nicht mehr in x vorkommt.
   Hinweis:
   Es ist empfehlenswert, für diese Programmieraufgabe die Schnittstelle (siehe oben) zunächst um die Methode `int` $pos(Zr\,x, Zr\,y)\{\ldots\}$ (Java) bzw. um die Funktionsprozedur `function` $pos(x : zr, y : zr)$ `:integer;` (Pascal) zu erweitern.
   Mit den Bezeichnungen

   $$x = X_1 X_2 \ldots X_i X_{i+1} \ldots X_{i+|y|} X_{i+|y|+1} \ldots X_{|x|};$$
   $$y = X_{i+1} \ldots X_{1+|y|}$$ (das erste Vorkommen von y in x, falls y in x überhaupt vorkommt)

   liefert *pos* den Index i, wenn y in x vorkommt, und sonst den Wert -1.

Fortsetzung nächste Seite!

2. x, y und z seien drei Wörter aus I^* wie bei 1.1

   2.1 Die Methode *ersetze* (aus 1.) terminiert sicher für alle x, wenn $|y| > |z|$ ist. Beweisen Sie diese Aussage!

   2.2 Geben Sie eine hinreichende Bedingung dafür an, dass die Methode *ersetze* (aus 1.) für alle x terminiert, wenn $|y| \leq |z|$ ist und wenigstens ein Zeichen aus I sowohl in y als auch in z vorkommt, und beweisen Sie Ihre Aussage.

Thema Nr. 2

Aufgabe 1 (Objektorientierung)

a) Definieren Sie die Begriffe *Klasse, Objekt, Vererbung, Interface* und *abstrakte Klasse*!

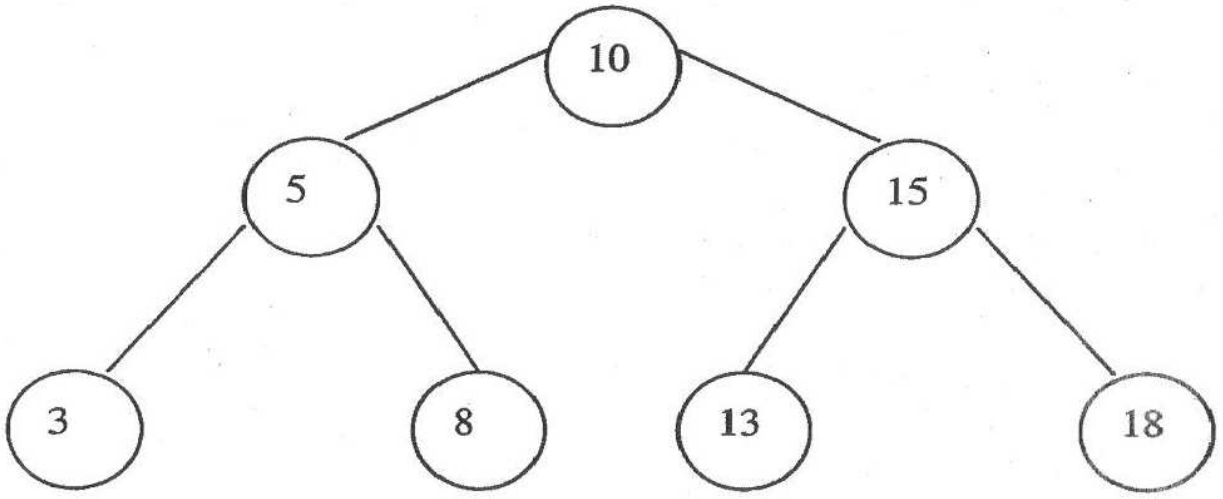
b) Schreiben Sie in Java (oder in einer anderen objektorientierten Programmiersprache) eine Klasse `Konto`, die das Guthaben eines Bankkontos verwaltet. Die Klasse soll Methoden zum `einzahlen` und `abheben` vom Konto haben. Das Konto soll man nicht überziehen können. Die Klasse soll außerdem eine Methode `getKontostand` haben, die den aktuellen Kontostand zurückliefert. Implementieren Sie den Konstruktor so, dass ihm das Startkapital des Kontos übergeben werden muss. Schreiben Sie ein Hauptprogramm, das die Klasse verwendet und zuerst ein Konto mit 100 € Startkapital eröffnet und dann 20 € abhebt.

c) Schreiben Sie eine von der Klasse `Konto` abgeleitete Klasse `Girokonto`. Im Gegensatz zu einem normalen Konto darf ein Girokonto um ein vorher festgelegtes Dispolimit überzogen werden. Schreiben Sie für die Klasse Girokonto Methoden zum Festlegen und Abfragen des Dispolimits! Der Einfachheit halber gehen Sie davon aus, dass für das Überziehen des Kontos keine Zinsen gezahlt werden müssen. Wenn nötig, überschreiben Sie Methoden der Klasse Konto.

Fortsetzung nächste Seite!

Aufgabe 2 (Binärer Suchbaum)

Gegeben sei der folgende binäre Suchbaum:



Löschen Sie den Schlüssel 13. Aus dem resultierenden Baum löschen Sie dann den Schlüssel 15. Im dritten Schritt löschen Sie dann den Schlüssel 5. Geben Sie dabei nach jeder Löschoperation den resultierenden Suchbaum an.

Aufgabe 3 (Code-Verständnis)

Unten ist der Java-Code einer Klasse `Stack` (ein Keller) abgedruckt, der alle Datenstrukturen und Methoden enthält, um einen Keller von natürlichen Zahlen zu verwalten. Die Klasse Stack besitzt eine Variable, die die Elemente des Kellers in einem Feld hält. Außerdem werden folgende Methoden nach außen zur Verfügung gestellt:

- `void init()`
  Initialisierung der Keller-Datenstruktur (mit leerem Keller zu Beginn)

- `boolean isEmpty()`
  Abfrage, ob der Keller gerade leer ist

- `void push(int i) throws StackException`
  soll die als Parameter angegebene Zahl auf den Keller legen; falls der Speicherbereich erschöpft ist, wird eine Ausnahmesituation ausgelöst.

- `int pop()`
  nimmt die oberste Zahl vom Keller und liefert sie zurück; ist der Keller leer, wird -1 zurückgegeben.

Fortsetzung nächste Seite!

Der folgende Code enthält allerdings einige syntaktische und semantische Fehler. Geben Sie jeweils die Zeilennummer des Fehlers und den korrigierten Code an.

```
1   void class Stack {
2      private int stackSize;
3      private int currentPos;
4      private int elements[];
5
6        Stack(int size) {
7           init();
8           if (size > 0) {
9              stackSize = size;
10             elements = new int[size];
11          }
12       }
13
14       // Initialisierung des Stacks
15       public void init() {
16          currentPos = 1;
17       }
18
19       // Ist der Stack leer?
20       public boolean isEmpty() {
21          return (currentPos = 0);
22       }
23
24       // Ablegen eines Wertes auf dem Stack
25       private void push(int i) {
26          if (currentPos == stackSize)
27             System.out.println ("Der Keller ist voll.");
28          else elements[currentPos++] = i;
29       }
30
31       // Holen des obersten Wertes vom Stack
32       public int pop() {
33          if (currentPos == 0)
34             return -1;
35
36          return (elements[--currentPos])
37       }
38    }
```

Fortsetzung nächste Seite!

Aufgabe 4 (Rekursion und Iteration)

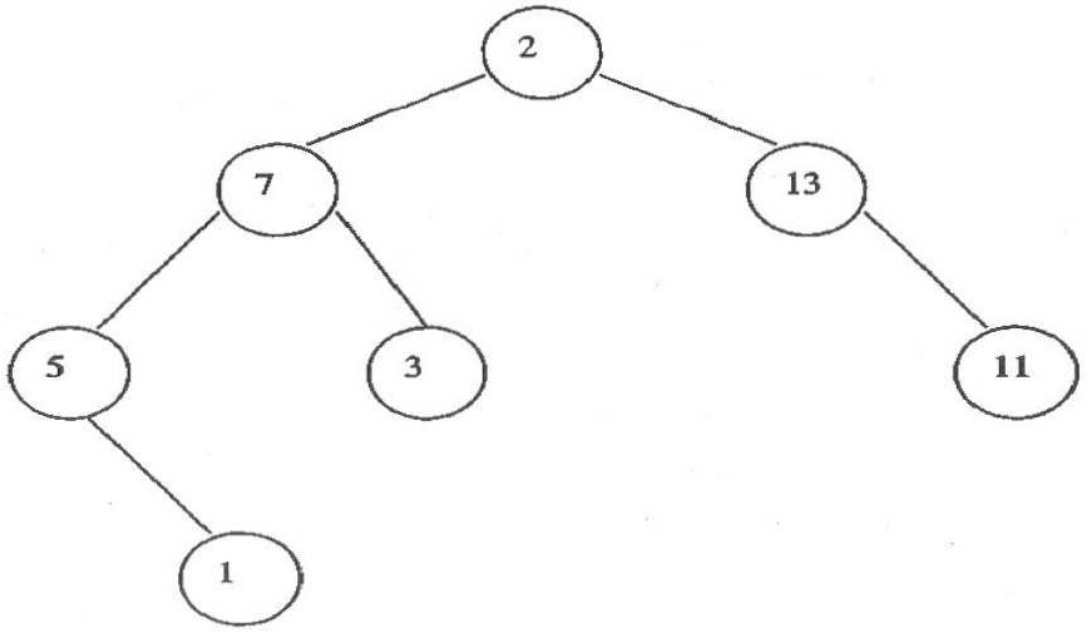
Die Potenzierungsfunktion für zwei natürliche Zahlen n, m kann durch fortgesetzte Multiplikation realisiert werden:

$$m^n = pot(n,m) = \begin{cases} m \cdot m^{n-1} & \text{falls } n > 0 \\ 1 & \text{sonst} \end{cases}$$

Geben Sie eine **iterative** und eine **rekursive** Implementierung der Potenzfunktion an! Benutzen Sie dabei keine Funktionen aus einer Bibliothek!

Aufgabe 5 (Rekursion und Iteration)

Gegeben sei der folgende Baum:



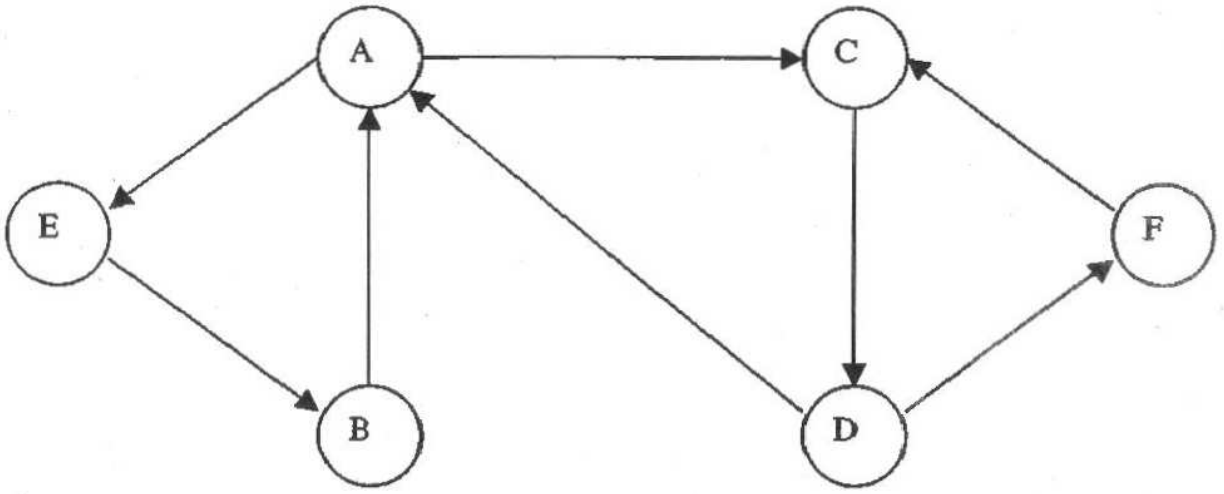
a) Durchlaufen Sie diesen Baum einmal in Tiefen- und einmal in Breitensuche und geben Sie jeweils an, in welcher Reihenfolge die im Baum enthaltenen Zahlen dabei ausgegeben werden!

b) Handelt es sich bei dem oben abgebildeten Baum um einen binären Suchbaum bzgl. der natürlichen Zahlen und der *kleiner*- Relation? Begründen Sie Ihre Antwort!

Fortsetzung nächste Seite!

Aufgabe 6 (Graphrepräsentation)

Repräsentieren Sie den folgenden Graphen sowohl mit einer Adjazenzmatrix als auch mit einer Adjazenzliste.



Aufgabe 7 (UML-Klassendiagramm)

Erzeugen Sie ein Analysediagramm (Klassendiagramm) für ein Informatiksystem einer Kart-Bahn, das die Vermietung der Kart-Wagen verwaltet. Folgende Anforderungen werden daran gestellt:

- Nur registrierte Kunden der Bahn können Kart-Wagen ausleihen. Jeder Kunde kann entweder einen oder sechs Karts ausleihen (zur Erklärung: bei sechs Karts möchte der Kunde für eine Gruppe mieten).
- Zu jedem Kunden werden Name, Adresse, Datum der ersten Miete und Kundennummer gespeichert.
- Zu jedem Kart ist das TÜV-Siegel (bescheinigt technisch korrekten Zustand des Karts), der Mietpreis, eine Inventarnummer und eine Typenbezeichnung festgehalten.
- Ein Kunde wählt zwischen der Miete von Karts für Erwachsene und solcher für Kinder.
- Zu einem Kart für Erwachsene wird die Leistung angegeben. Karts für Kinder haben alle dieselbe reduzierte Leistung.

Erstellen Sie ein UML-Klassendiagramm für die Analyse, das ausschließlich mit den Klassen Kunde, Kart, KartErwachsene und KartKinder arbeitet und das die gestellten Anforderungen erfüllt. Kardinalitäten/Multiplizitäten geben Sie bitte auch im Fall einer Kardinalität/ Multiplizität von 1 explizit an.

Fortsetzung nächste Seite!

Aufgabe 8 (Algorithmenverständnis)

Analysieren Sie nachfolgenden Pseudocode eines bekannten Algorithmus.

```
1     op1( k, x ):
2          vater := null;
3          sohn  := k.root;
4          while not (sohn = null) do
5               vater := sohn;
6               if x.key < sohn.key
7                    then sohn := sohn.left
8                    else sohn := sohn.right
9               fi
10         od
11         if vater = null
12              then k.root := x
13              else
14                   if x.key < vater.key
15                        then vater.left := x
16                        else vater.right := x
17                   fi
18         fi
```

a) Erläutern Sie kurz, was in jeder Zeile geschieht!

b) Um welchen Algorithmus handelt es sich hier?

c) Gehen Sie davon aus, dass zu Beginn k.root=null ist. Wählen Sie fünf Werte für x aus und wenden Sie den Algorithmus nacheinander auf diese Werte an! Geben Sie hierzu nach jedem der fünf Durchläufe das Ergebnis an!

Aufgabe 9 (Textformatierung)

Schreiben Sie ein Programm in einer Programmiersprache Ihrer Wahl, das ein zweidimensionales Integer-Feld (Array) der Größe 5×5 auf dem Bildschirm ausgibt. Jede Zeile des Feldes soll in einer eigenen Zeile der Ausgabe erscheinen. Die Zahlen innerhalb der Zeile des Feldes sollen am Bildschirm mit einem Komma voneinander getrennt sein. Nach der letzten Zahl einer Zeile soll kein Komma stehen.

Beispiel: Das Java-Feld

```
int [][] a =   { {1,2,3,4,5}, {6,7,8,9,10}, {11,12,13,14,15},
                 {16,17,18,19,20}, {21,22,23,24,25}};
```

soll wie folgt formatiert ausgegeben werden:

```
1,2,3,4,5
6,7,8,9,10
11,12,13,14,15
16,17,18,19,20
21,22,23,24,25
```