Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst

2000

46114

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: **Informatik (nicht vertieft studiert)**

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

Anzahl der gestellten Themen (Aufgaben): 2

Anzahl der Druckseiten dieser Vorlage: 5

Bitte wenden!

Thema Nr. 1

1. Algorithmen und Datenstrukturen

Ein binärer Baum ist entweder leer, oder er besteht aus einer Wurzel und zwei binären Bäumen als linkem und rechtem Unterbaum.

Ein binärer Suchbaum für die Schlüsselmenge $S = \{s_1, s_2, \ldots, s_n\}$ ist ein binärer Baum mit n Knoten $\{ v_1, v_2, \ldots, v_n\}$ und einer bijektiven Beschriftung $c(v)$: $V \rightarrow S$ der Knoten, so dass für jeden Knoten v folgende Eigenschaft erfüllt ist: Für alle Knoten v' im linken Unterbaum und alle Knoten v'' im rechten Unterbaum gilt: $c(v') < c(v) < c(v'')$.

(1a) Beschreiben Sie verbal, ggf. unter Zuhilfenahme einer Skizze, das Suchen nach einem Knoten mit vorgegebenem Schlüssel!
Woran erkennt der Algorithmus, dass kein Element mit diesem Schlüssel im Baum enthalten ist?
Welche charakteristische Größe des Baumes bestimmt die Suchzeit?

(1b) Beschreiben Sie genauso das Einfügen eines neuen Knotens, der durch einen bisher nicht benutzten Schlüssel charakterisiert ist!

(1c) Die Reihenfolge, in der Knoten in einen ursprünglich leeren Suchbaum eingetragen werden, kann wesentlich die spätere Suchzeit beeinflussen.
Welche beiden Extremfälle können auftreten, und was kann man in beiden Fällen über die Suchzeit sagen?

(1d) Soll man einen Knoten mit vorgegebenem Schlüssel aus dem Baum entfernen, so wird er zuerst gesucht. Handelt es sich um einen Knoten mit maximal einem nichtleeren Unterbaum, so kann er einfach entfernt werden.
Was ist zu tun, wenn der zu entfernende Knoten zwei nichtleere Unterbäume besitzt?

2. Programmiermethodik

Mit folgender Datenstruktur können in C binäre Suchbäume dargestellt werden:

```
struct Node {
  struct Node *left, *right;  /* linker bzw. rechter Unterbaum */
  int    key;                 /* Sortierschluessel */
  info   data;                /* zugehoerige Information */
};
```

oder in PASCAL:

```
TYPE nodepointer = ^node;
     node = RECORD left, right: nodepointer;
                   key:         integer;
                   data:        info
            END;
```

(Der Datentyp `info` sei geeignet definiert.)

Fortsetzung nächste Seite!

(2a) Ein typisches Strukturierungsmodell bei der Programmentwicklung im Kleinen ist die Rekursion.
Erläutern Sie diese verbal anhand der Suche nach einem Knoten mit vorgegebenem Schlüssel in einem binären Suchbaum!

(2b) Schreiben Sie diesen Suchvorgang in C oder PASCAL auf!
(Bemerkung: Sie dürfen C durch C++ oder Java, PASCAL durch MODULA oder OBERON ersetzen, müssen dann aber die oben angegebene Datenstruktur neu definieren.)

(2c) Wenn Sie das Suchprogramm günstig geschrieben haben, ist es ein Beispiel für die sogenannte Endrekursion.
Was versteht man darunter, und was ist ihr Vorteil?

(2d) Schreiben Sie (in der von Ihnen gewählten) Programmiersprache eine rekursive Prozedur, die die im Baum auftretenden Schlüssel als Reihung in aufsteigender Folge abspeichert!
Leiten Sie das Konzept Ihres Programmes zunächst aus der Definition des binären Suchbaumes her!

3. Systementwurf

Wir betrachten ein einfaches Beispiel für einen datenstrukturorientierten Programmentwurf.

(3a) Die folgende formale Grammatik in Backus-Naur-Form (BNF) beschreibt das prinzipielle Aussehen einer Rechnung:

Rechnung	::=	*Datum Rumpf Gesamtsumme*
Rumpf	::=	*Position* | *Rumpf Position*
Position	::=	*Menge Bezeichnung Einzelpreis Gesamtpreis*
	|	1 *Bezeichnung Gesamtpreis*

Erläutern Sie die verwendeten Elemente der BNF!
Woran liegt es, dass man mit einer endlichen Beschreibung einen unendlichen Vorrat an Rechnungen beschreiben kann?

(3b) Die erweiterte Backus-Naur-Form (EBNF) kennt die Stern-Operation.
Verwenden Sie diese, um aus der zweiten Zeile des Beispiels die Rekursion zu entfernen!

(3c) Entwickeln Sie aus der EBNF-Beschreibung des Beispiels unmittelbar eine Programmstruktur, z.B. als Struktogramm!

(3d) Fassen Sie allgemein zusammen, wie man aus einer mit EBNF beschriebenen Datenstruktur unmittelbar die Programmstruktur erzeugen kann! (Geben Sie zu jeder Konstituenten der EBNF die entsprechende programmiersprachliche Konstruktion an!)

Thema Nr. 2

Für die Realisierung der Programme in den folgenden Aufgaben können Sie eine beliebige höhere Programmiersprache wählen. Geben Sie die gewählte Programmiersprache an! Für alle Aufgaben ist die gleiche Programmiersprache zu wählen. Kommentieren Sie Ihre Programme so gut wie möglich!

Aufgabe 1

a) Erläutern Sie die Parameterübergabemechanismen Call-by-value und Call-by-reference!

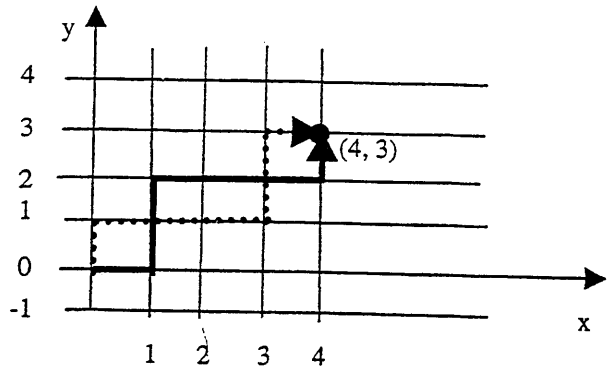
b) Was sind programmiermethodisch die Vor- und Nachteile dieser Parameterübergabemechanismen für die Übergabe von Grunddatentypen wie `INTEGER` und von strukturierten Datentypen wie z.B. Felder und Verbunde?

c) Gegeben sei eine Operation f mit einem formalen Parameter a vom Typ `ARRAY OF INTEGER`. Erklären Sie, ob und wie die Wahl des Parameterübergabemechanismus Einfluss auf die Zeit- und Speicherplatzkomplexität der Implementierung von f haben kann!

Aufgabe 2

a) Erklären Sie die Begriffe rekursive Funktion und rekursiver Datentyp!

b) Gegeben sei ein zweidimensionales Raster mit ganzzahligen Koordinaten. Ein Weg zwischen zwei Rasterpunkten verlaufe nur auf dem Raster und zwar für jedes Teilstück in Laufrichtung von links nach rechts oder von unten nach oben.



Die Abbildung zeigt zwei mögliche Wege vom Ursprung (0, 0) zum Punkt (4, 3).
Schreiben Sie in einer höheren Programmiersprache ein rekursives Programm

```
alleWege(i: INTEGER, j: INTEGER): INTEGER
```

das die Anzahl aller möglichen Wege vom Ursprung (0, 0) zum Punkt (i, j) berechnet!

c) Beweisen Sie, dass Ihr Programm für alle ganzen Zahlen i, j terminiert!

Fortsetzung nächste Seite!

Aufgabe 3

a) Definieren Sie in einer höheren Programmiersprache einen Datentyp `Farbpunkt` für farbige Punkte im zweidimensionalen Raum! Ein farbiger Punkt besitze die Koordinaten `x_coord`, `y_coord` und eine Farbe. Als Farben seien nur `rot` und `schwarz` zugelassen.

b) Schreiben Sie ein Programm für die Operation

```
kleiner(p: Farbpunkt, q: Farbpunkt): Boolean
```

das genau dann *true* liefert, wenn p einen kleineren (euklidischen) Abstand vom Ursprung hat als q!

c) Schreiben Sie ein Programm für die Operation

```
minSchwarz (a: ARRAY OF Farbpunkt): Farbpunkt
```

das als Resultat einen kleinsten schwarzen Punkt aus a liefert, sofern a mindestens einen schwarzen Punkt enthält!

d) Welche Möglichkeiten kennen Sie, um `minSchwarz` robust zu gestalten, so dass `minSchwarz` auch dann ein aussagekräftiges Ergebnis liefert, wenn a keinen schwarzen Punkt enthält?

Aufgabe 4

Ein Autokonzern investiert für die Entwicklung eines neuen Modells einen Betrag *I*. Der Konzern möchte berechnen, nach wie vielen Jahren sich der Investitionsbetrag amortisiert hat. Dabei wird von folgenden fest gegebenen Daten ausgegangen:

- Herstellungskosten für ein Auto im Jahr: 22500,00 DM,
- Erhöhung der Herstellungskosten nach jedem Jahr um 1,5 %,
- Verkaufspreis für ein Auto in den ersten drei Jahren: 25000,00 DM,
- Erhöhung des Verkaufspreises nach jeweils drei Jahren um 7,0 %.

Der Reingewinn eines verkauften Autos ist die Differenz des Verkaufspreises und der Herstellungskosten (im aktuell betrachteten Jahr). Der Konzern geht davon aus, dass jedes Jahr eine gleich bleibende Anzahl *Anz* von Autos verkauft wird.

Schreiben Sie ein Programm, das den Investitionsbetrag *I* und die Anzahl *Anz* der pro Jahr verkauften Autos einliest und daraufhin berechnet und ausgibt, nach wie vielen Jahren der durch den Verkauf des neuen Modells insgesamt entstandene Reingewinn den Investitionsbetrag erreicht oder übertroffen hat!