Institut 186 für Computergraphik	**Algorithmen und Datenstrukturen 1**			1. Übungstest 24. April 1998 Gruppe A	
Matrikelnr.	**Beil.**†	1 (8)	2 (16)	3 (14)	4 (12)
NACHNAME, Vorname					Σ (50)

† Geben Sie an, wieviele Zusatzbl. Sie abgeben (**jedes mit Name & Matr.-Nr. beschriftet!**).

(8) Aufgabe 1 – Aufwand

Setzen Sie zwischen die beiden Os jeweils das gültige Vergleichszeichen ($=$, $<$ oder $>$).

(2) **a)**

$$O\left(N^3 - N^2 + N - 1\right) \qquad O\left(\frac{N!}{(N-3)!}\right)$$

(2) **b)**

$$O\left(\sqrt[5]{N^4 - N^2} \cdot \sqrt[3]{N^3 - 1}\right) \qquad O\left(((N^2 - 1) \cdot (N^2 + 1))^{\frac{1}{2}}\right)$$

(2) **c)**

$$O\left(N \cdot log(N^2) \cdot (N^2 \cdot log(N) - 1)\right) \qquad O\left((N \cdot ln(N) - 1)^3\right)$$

(1) **d)**

$$O\left(ln(N^3)\right) \qquad O\left((log(N))^3\right)$$

(1) **e)**

$$O\left(\pi \cdot N^2\right) \qquad O\left(e + \sqrt{N^4 + N^2}\right)$$

(16) Aufgabe 2 – Allgemeiner Baum

Die Filenamen in einem hierarchischen Filesystem sind in einem allgemeinen Baum gespeichert, das heißt, jeder Knoten entspricht einem File oder einem Directory. Wenn der Knoten einen Nachfolger (`subTree`) hat, so handelt es sich um ein Directory, wenn `subTree` gleich `nil` ist so handelt es sich um ein File. Die Files innerhalb eines Directories sind *nicht sortiert*. Die dazugehörige Datenstruktur sieht folgendermaßen aus:

```
TYPE
    NodePtr = ^Node;
    Node = RECORD
             fileName:   STRING;
             next:   NodePtr;
             subTree: NodePtr;
           END;
```

a) Schreiben Sie eine *effiziente, rekursive* Funktion

```
PROCEDURE find(path: STRING;
               filesystem : NodePtr;
               name : STRING);
```

die all diejenigen Files mit dem Pfad zu ihnen ausgibt, deren Namen `name` ist. Geben Sie jedoch nur Pfade aus, deren Gesamtlänge (inkl. Filenamen und Trennzeichen) kleiner als 80 Zeichen ist.

Verwenden Sie für Ihren Algorithmus *keine globalen Variablen!*

Hinweis: Sie können Strings mit `=` vergleichen, mit der `FUNCTION length(str:STRING):INTEGER` erfahren Sie die Länge von Strings und die Prozedur:

```
PROCEDURE append(toString: STRING; otherString:STRING);
```

hängt `otherString` an `toString` an.

Beispiel: Gegeben sei folgendes Filesystem:

```
MyDir/
    FileOne
    MySubDir/
        FileOne
    FileTwo
```

Die Suche nach '`FileOne`' soll folgende Ausgabe liefern:

```
MyDir/FileOne
MyDir/MySubDir/FileOne
```

b) Geben Sie außerdem noch den Aufruf einer Suche nach dem File '`core`' in einem Filesystem mit dem Namen (`NodePtr`) `unixFileSystem` an.

(14) Aufgabe 3 – Sortiere Listen

Gegeben sei eine Datenstruktur für sortierte Listen von ganzen Zahlen.

```
TYPE NodePtr = ^Node;
     Node = RECORD
              value: INTEGER;
              next: NodePtr;
            END;
```

a) Schreiben Sie eine effiziente Funktion:

```
FUNCTION merge(list1, list2 : NodePtr) : NodePtr;
```

die aus den zwei aufsteigend sortierten Listen *list1* und *list2* eine einzige aufsteigend sortierte Liste macht und diese zurückliefert.

Sie sollen dabei die Knoten der übergebenen Liste verwenden und *keine* Knoten *löschen* oder *erzeugen*! Beachten Sie auch alle Sonderfälle!

Hinweis: Betrachten Sie die ersten Knoten der beiden übergebenen Listen. Wählen Sie den Knoten mit der kleineren Zahl und hängen Sie ihn ans Ende der Ergebnisliste. Wiederholen Sie solange bis keine Knoten mehr übrig sind.

b) Die beiden Eingabelisten für `merge` haben eine Länge von N bzw. M Knoten. Was ist der Aufwand dieser Funktion in O-Notation, wenn Sie effizient programmiert wurde?

(12) Aufgabe 4 – Hashtabelle

a) Eine Hashtabelle soll die Patienten eines Arztes mit ihrer Patientennummer als Schlüssel speichern. Die Hashfunktion h ist gegeben und berechnet den Hashwert als **Ziffernsumme der Patientennummer modulo 7**. Tragen Sie die folgenden Patienten in der angegebenen Reihenfolge in eine Hashtabelle mit äußerer Verkettung und zeichnen Sie danach die Hashtabelle:

```
4346   Meyer
2506   Schmidt
7643   Nagler
9459   Kogelbauer
1526   Bauer
2412   Hader
6341   Niedermayer
3033   Anderl
```

b) Die Patienten werden nun in eine zweite Hashtabelle eingetragen, diesmal mit dem Namen als Schlüssel. Als Hashfunktion h dient diesmal die **Länge des Namens modulo 7**. Die Patienten werden wieder in der oben angegebenen Reihenfolge eingetragen. Zeichen Sie danach die beiden Hashtabellen in der selben Darstellungsart wie Abbildung 5.4 im Skriptum auf Seite 103. **Die Hashtabelle der Patientennummern soll dabei auf der linken Seite sein.**