Institut 186 für Computergraphik	**Algorithmen und Datenstrukturen 1**			Beispieltest 27. Juni 1997 Gruppe A
Matrikelnr.	**Beil.**†	1 (20)	2 (21)	3 (9)
Nachname, Vorname				Σ (50)

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

(20) Aufgabe 1 – Sparse Matrizen als Quadtree (1. Beispiel)

In einem Programm werden spärlich besetzte Matrizen als Quadtrees repräsentiert. Hierzu werden folgende Datenstrukturen verwendet:

```
QuadTree = ^QuadNode;
QuadNode =  OBJECT
              FUNCTION isLeaf : BOOLEAN; VIRTUAL;
              FUNCTION add(otherTree: QuadTree) : QuadTree; VIRTUAL;
              FUNCTION subNode(i : 1..4) : QuadTree; VIRTUAL;
              FUNCTION getValue : FLOAT; VIRTUAL;
            END;
LeafNode  = OBJECT(QuadNode)
              value: REAL;
              CONSTRUCTOR Init(new_value: REAL);
              FUNCTION isLeaf: BOOLEAN;  (* immer TRUE *)
              FUNCTION subNode(i : 1..4) : QuadTree; (* immer NIL *)
              FUNCTION getValue : FLOAT; (* returns value *)
            END;
SplitNode = OBJECT(QuadNode)
              subNodes: ARRAY [1..4] OF QuadTree;
              CONSTRUCTOR Init(new_subNodes: ARRAY[1..4] OF QuadTree);
              FUNCTION isLeaf: BOOLEAN;  (* immer FALSE *)
              FUNCTION subNode(i : 1..4) : QuadTree;
                  (* accessor method for subNodes[1..4] *)
              FUNCTION getValue : FLOAT; (* error when called *)
            END;
```

Implementieren Sie die beiden Methoden die notwendig sind, um zwei als Quadtree repräsentierte Matrizen zu addieren:

```
FUNCTION SplitNode.add(otherTree: QuadTree) : QuadTree;
FUNCTION LeafNode.add(otherTree: QuadTree) : QuadTree;
```

Hierbei ist zu beachten, daß die resultierende Matrix neu angelegt und als Funktionswert zurückgeliefert werden soll. Desweiteren soll beachtet werden, daß vier Knoten zusammegefasst werden, sobald sie den gleichen Wert haben. Vorsicht, das kann mehrere Ebenen direkt hintereinander betreffen! Sie können davon ausgehen, daß die übergebenen QuadTrees valid, d.h. ungleich NIL sind.

(21) Aufgabe 2 – Tabellenkalkulation (6. Beispiel)

Eine einfache Tabellenkalkulation, die nur das Addieren von zwei referenzierten Feldern, und das Eintragen eines direkten Zahlenwertes erlaubt, sei gegeben. Die dazu notwendigen Datenstrukturen sehen folgendermaßen aus:

```
TYPE
  ReferencePtr = ^Reference;
  Reference = RECORD
                row, column: INTEGER;
                next: ReferencePtr;
              END;
  Cell = RECORD
           value: REAL;
           operands: ReferencePtr;      (* Backpointers *)
           forward: ReferencePtr;       (* Forwardpointers *)
         END;
 VAR table: ARRAY [1..MAXROW,1..MAXCOLUMN] OF Cell;
```

Bei einer nicht berechneten Zelle ist `operands` gleich `NIL`, bei einer berechneten Zelle ist `operands` eine Liste mit zwei Elementen, die auf die beiden Operanden verweisen. `forward` ist eine Liste mit Referenzen auf alle abhängigen Zellen. Sie können annehmen, daß in den Abhängigkeiten keine Zyklen vorkommen. Sowohl `operands` als auch `forward` sind schon korrekt gesetzt. Ein alternativer Algorithmus zum Neuberechnen aller von einer veränderten Zelle abhängigen Zelle funktioniert folgendermaßen: zuerst berechnet man den neuen Wert der veränderten Zelle, danach berechnet man die neuen Werte aller direkt abhängigen Zellen, danach berechnet man die neuen Werte aller einfach indirekt abhängigen Zellen, usw.
Schreiben sie eine *rekursive* Funktion

```
FUNCTION recalculatelevel(level,row,column: INTEGER) : BOOLEAN;
```

die alle von der angegebenen Zelle abhängigen Zellen, neu berechnet, wobei `level` den Grad der Indirektion angibt, d.h. wenn `level = 0` ist, wird der Wert der Zelle neu berechnet, wenn `level = 1` ist wird nur der Wert aller direkt abhängigen Zellen neu berechnet, wenn `level = 2` werden nur alle einfach indirekt abhängigen Zellen neu berechnet, usw.. Der Rückgabewert der Funktion soll genau dann `TRUE` sein, wenn mindestens eine Neuberechnung durchgeführt wurde. Um Mehrfachberechnungen zu vermeiden, verwenden Sie folgendes Feld:

```
VAR visited: ARRAY [1..MAXROW,1..MAXCOLUMN] OF BOOLEAN;
```

von dem Sie annehmen können, daß alle Elemente mit `FALSE` initialisiert sind. Schreiben Sie nun die *effiziente* Prozedur

```
PROCEDURE recalculate(row,column: INTEGER);
```

die `recalculatelevel` verwendet um den alternativen Algorithmus vollständig zu implementieren.

(9) Aufgabe 3 – Diverses

a) Gegeben ist die Bibliotheksverwaltung aus dem 5. Beispiel. Es soll zusätzlich zu den angegebenen Operationen auch noch möglich sein einen nach ISBN Nummern sortierten Katalog der Bücher zu erstellen. Ist die gewählte Datenstruktur (Hashtable) für diese Operation geeignet? Welche zusätzliche Datenstruktur würden Sie für diese Operation vorschlagen?

b) Gegeben ist der Landkartenfärbealgorithmus aus dem 7. Beispiel. Ein Programmierer kommt auf die Idee, einen binären Suchbaum zu verwenden, um jeweils das Land mit den meisten ungefärbten Nachbarn zu finden. Ist das besser oder schlechter als die im Beispiel vorgeschlagene Datenstruktur?

c) Gegeben ist das WATERWORLD Programm aus dem 9. Beispiel. Geben Sie zu jeder der 6 Fähigkeiten eines Lebewesens (simuliert werden, vermehren, dargestellt werden, eigelesen werden, eingegeben werden, auf File geschrieben werden) an, ob die jeweilige Methode des Objekts `LEBEWESEN` virtuell oder nicht virtuell ist (Keyword: `VIRTUAL`).

Achtung: Begründung sie alle Ihre Antworen! Ein richtiges Resultat oder eine richtige Antwort wird **nicht gewertet**, wenn aus Ihrer Antwort nicht hervorgeht, wie Sie zu dem Ergebnis oder dem Schluß gekommen sind.