Kurs 1578 HK

Aufgabe 1 (1 + 1 + 8 + 8 + 2 = 20 Punkte)

Wir betrachten einfache rechteckige Wandmosaike aus quadratischen bunten Fliesen. Jede einzelne Fliese möge eine oder mehrere Farben aus dem Wertebereich eines Aufzählungstyps `tFarbe` haben. Wir wollen die Mosaike als zweidimensionale Felder von Farbmengen mit `integer`-Zahlen als Spalten- und Zeilenindizes darstellen, wobei wir die Spalten von links nach rechts und die Zeilen von unten nach oben (!) zählen.

Wir sagen, eine Fliese *schimmere* in der Farbe F, wenn F zu den Farben der Fliese selbst oder zu denen der Nachbarfliese rechts oben von ihr gehört.

Beispiel:

Rot	Rot	Rot	Rot Gelb
Blau	Blau	Blau	Rot
Rot Blau	Rot Blau	Gelb Blau	Rot

In diesem Mosaik schimmert zum Beispiel die Fliese in der zweiten Zeile und dritten Spalte in Blau, Rot und Gelb. Gleiches gilt für die Fliese in der ersten Zeile und dritten Spalte. Die Fliese in der dritten Zeile und dritten Spalte schimmert nur in Rot. In Rot schimmern in diesem Mosaik sogar alle Fliesen.

(a) Definieren Sie einen set-Typ `tFarbmenge` für Mengen von Farben aus dem Wertebereich von `tFarbe`.

(b) Definieren Sie einen geeigneten Typ `tMosaik` für die Darstellung von Mosaiken mit 30 Zeilen und 20 Spalten und Farbmengen des Typs `tFarbmenge`.

(c) Schreiben Sie eine Funktion `schimmert`, die zu einem als Parameter zu übergebenden Mosaik des Typs `tMosaik` aus (b) prüft, ob es eine Farbe gibt, in der jede Fliese des Mosaiks schimmert (in obigem Beispiel war dies - nämlich für die Farbe Rot - der Fall). Hinweis: Wir verlangen nicht, daß Sie Schleifen abbrechen, sobald das Prüfergebnis feststeht.

(d) Schreiben Sie eine Prozedur `bestimmeFarben`, die zu einem als Parameter zu übergebenden Mosaik die Menge aller in ihm vorkommenden Farben bestimmt. Lassen Sie hier als Mosaike Felder mit beliebigen Ausschnittstypen von `integer` als Indextypen zu (Stichwort: Array-Gerüste).

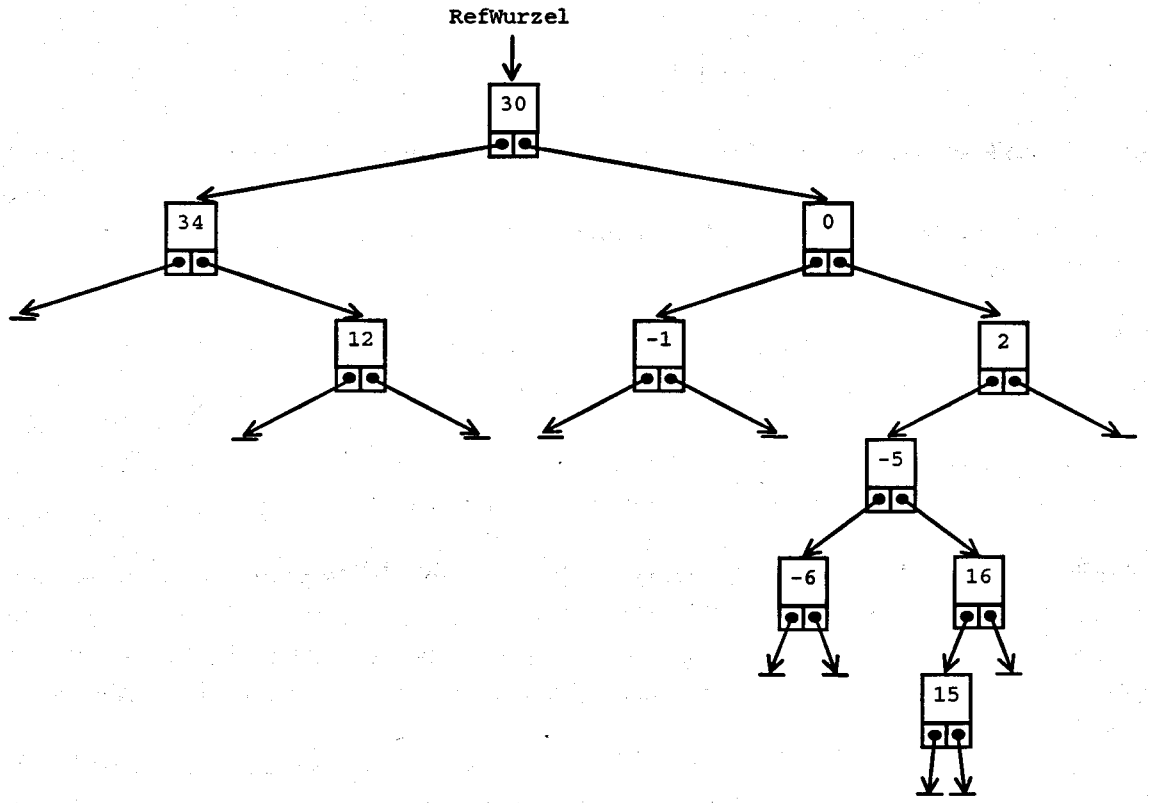
(e) Definieren Sie einen geeigneten Aufzählungstyp `tFarbe` für den speziellen Fall, daß in den Mosaiken die Farben Rot, Grün, Gelb, Blau, Violett, Weiß, Grau und Schwarz vorkommen können, und geben Sie für diesen speziellen Fall einen möglichst kurzen Ausdruck an für die Menge aller Farben aus seinem Wertebereich.

Aufgabe 2 (1 + 4 + 10 = 15 Punkte)

Wir betrachten binäre Bäume von ganzen Zahlen im Wertebereich von `integer`.

Ein *binärer Baum* ist wie eine lineare Liste entweder leer oder enthält genau ein erstes Element, ein Element ohne Vorgänger, bei Bäumen *Wurzel* genannt. Anders als eine lineare Liste verzweigt ein binärer Baum jedoch bei jedem Element in zwei wohlunterschiedene Richtungen (*links* und *rechts*), so daß jedes Element entweder keinen unmittelbaren Nachfolger besitzt oder einen linken oder einen rechten oder einen linken und einen rechten. Die Elemente nennt man auch *Knoten*, die unmittelbaren Nachfolger *Söhne*. Wie bei einer linearen Liste sind alle Elemente außer dem ersten unmittelbar oder mittelbar Nachfolger des ersten Elements und haben genau einen unmittelbaren Vorgänger - dies garantiert die an die Verzweigung natürlicher Bäume erinnernde Struktur. Jedes Element eines binären Baumes bildet zusammen mit seinen Nachfolgern wieder einen binären Baum (*Teilbaum*), dessen Wurzel es ist.

Beispiel:

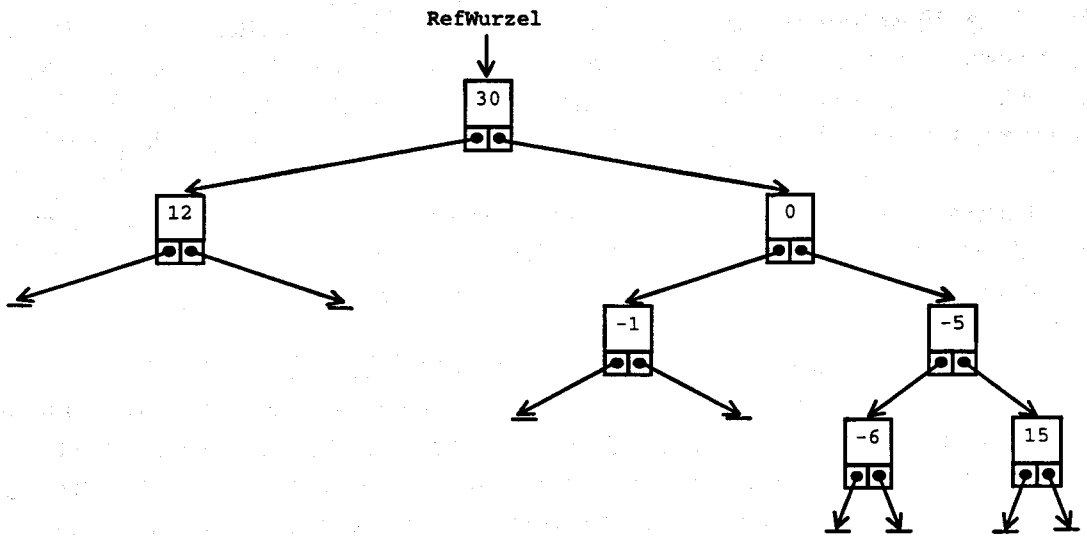


(a) Definieren Sie einen geeigneten Typ für solche binären Bäume.

(b) Schreiben Sie eine rekursive Prozedur `loescheBinBaum`, die den binären Baum löscht, auf dessen Wurzel der als Parameter zu übergebende Zeiger zeigt. Der Speicherplatz der Knoten soll freigegeben werden. Der übergebene Zeiger darf nach dem Aufruf der Prozedur undefiniert sein.

(c) Schreiben Sie eine rekursive Prozedur `loescheUnverzweigte`, die in dem binären Baum, auf dessen Wurzel der als Parameter zu übergebende Zeiger zeigt, alle diejenigen Knoten entfernt, die genau einen Sohn haben. Der Speicherplatz der entfernten Knoten soll freigegeben werden.

Beispiel: Nach dem Aufruf `loescheUnverzweigte (RefWurzel)` wird der Baum aus obigem Beispiel zu



Aufgabe 3 (18 Punkte)

Wir wollen einfach verkettete lineare Listen paarweise verschiedener, nicht-negativer ganzer Zahlen aus dem Wertebereich von `integer` bearbeiten. Eine Teilliste von aufsteigend geordneten Zahlen, die in der Ausgangsliste unmittelbar aufeinander folgen, wollen wir hier *Sequenz* nennen. Definieren Sie die hierfür benötigten Typen, und schreiben Sie eine Prozedur, die zu einer solchen Liste eine längste Sequenz ermittelt. Zurückgegeben werden sollen ein Zeiger auf den Anfang der Sequenz und die Länge (d.h. die Zahl der Elemente) der Sequenz.

Beispiel: Die Liste 1 → 0 → 19 → 54 → 7 → 10 → 15 → 11 → 20 enthält zwei längste Sequenzen: 0 → 19 → 54 und 7 → 10 → 15; von der Prozedur zurückgegeben werden sollen ein Zeiger auf die 0 oder ein Zeiger auf die 7 sowie die Länge 3.

Aufgabe 4 (6 + 26 + 6 + 11 + 3 = 52 Punkte)

In einer Datei seien Informationen zu allen Einsendungen eines Semesters zu Kursen des Fachbereichs Informatik gespeichert*). Zu jeder Einsendung enthalte die Datei jeweils die Matrikelnummer des Einsenders (1-30000), die Kursnummer (1500-1999), die Kurseinheit (1-7) und die erreichte Punktzahl (0-100). Über die Reihenfolge der Einsendungen in der Datei sei nichts bekannt.

Wir wollen diese Einsendungen in einer anderen Datenstruktur sortiert verbuchen und anschließend auswerten. Dazu soll als „Gesamtergebnis“ des Semesters zu jedem der Kurse 1500 bis 1999 eine einfach verkettete lineare Liste angelegt werden, in die - nach Matrikelnummern aufsteigend geordnet - zu jedem Einsender ein Element eingefügt wird. In dieses Element sollen seine Matrikelnummer und zu allen Kurseinheiten 1 bis 7 des betreffenden Kurses die von ihm zu dieser Kurseinheit erreichte Punktzahl eingetragen werden (0 Punkte, falls er zu dieser Kurseinheit nichts eingesendet hat). Wer zum Kurs gar nichts eingesendet hat, soll in der Liste auch nicht erscheinen; gab es keine Einsender, bleibt die Liste also leer.

(a) Definieren Sie die hierfür benötigten Typen, darunter einen Typ `tEinsendung` für die Komponenten der Datei, einen Typ `tEinsendedatei` für die Datei, einen Typ `tErgebnis` für die Listenelemente und einen Typ `tGesamtergebnis` für die neue Datenstruktur mit dem sortierten Gesamtergebnis. Setzen Sie dabei die folgenden Konstantendefinitionen voraus (einige der Konstanten werden erst weiter unten benötigt):

```
const
MAXMATRIKELNR  = 30000;
MINKURSNR      = 1500;
MAXKURSNR      = 1999;
MAXKURSEINHEIT = 7;
MAXPUNKTE      = 100;
MINERFOLG      = 200;    { Mindestpunktzahl fuer Erfolg }
MAXPUNKTSUMME  = 700;    { MAXPUNKTE * MAXKURSEINHEIT }
```

(b) Schreiben Sie eine Prozedur `verbucheEinsendung`, die die Informationen des Typs `tEinsendung` zu einer einzelnen Einsendung in einem sortierten Gesamtergebnis des Typs `tGesamtergebnis` verbucht. Sollte im Gesamtergebnis zu Kurs, Einsender und Kurseinheit bereits eine Punktzahl eingetragen sein, soll sie überschrieben werden.
Hinweis: Achten Sie auf die Ordnung nach Matrikelnummern. Zur selben Matrikelnummer wie in der zu verbuchenden Einsendung kann es in der zum Kurs gehörenden Liste bereits einen Eintrag geben oder nicht, ebenso zu kleineren Matrikelnummern. Die Liste kann auch noch leer sein. Nehmen Sie sich Zeit für die Strategie zum Auffinden der richtigen Stelle für die Einfügung! Beachten Sie, daß in Standard-Pascal boolesche Ausdrücke stets vollständig ausgewertet werden, und führen Sie geeignete boolesche Variablen ein.

(c) Schreiben Sie unter Benutzung der Prozedur `verbucheEinsendung` aus (b) eine Prozedur `verbucheAlleEinsendungen`, die aus allen Einsendungen einer Datei des Typs `tEinsendedatei` ein sortiertes Gesamtergebnis des Typs `tGesamtergebnis` erzeugt.

---

*) Um die Schreibarbeit zu verringern, wollen wir annehmen, daß aufgund eines Frauenförderplans die Einsendungen weiblicher Studierender in einer gesonderten Datei gespeichert werden, so daß die hier zu bearbeitende Datei ausschließlich Einsendungen männlicher Studierender enthält.

(d) Ein Einsender zu einem Kurs gelte als erfolgreich, wenn die über alle Kurseinheiten des Kurses summierte Punktzahl seiner Einsendungen (Punktsumme) mindestens 200 Punkte betrug. Schreiben Sie eine Prozedur `gibErfolgreicheAus`, die zu einem Gesamtergebnis des Typs `tGesamtergebnis` die Punktsummen aller erfolgreichen Einsender über die Standardausgabe ausgibt. Die Ausgabe soll nach Kursnummern aufsteigend geordnet zu jedem Kurs, zu dem es Einsendungen gab (und nur zu solchen) eine nach Matrikelnummern aufsteigend geordnete Tabelle enthalten, in der für jeden Einsender, der zu diesem Kurs erfolgreich war, seine Matrikelnummer und die im Kurs erreichte Punktsumme erscheint.

Die Ausgabe soll folgende Form haben:

```
Erfolgreich waren

zu Kurs 1500
Matrikelnr. Punkte
     732     699
    1112     200
   23027     425

zu Kurs 1578
Matrikelnr. Punkte

zu Kurs 1815
Matrikelnr. Punkte
    1112     496
    5819     513
```

(In diesem Beispiel gab es zu Kurs 1578 zwar Einsender, aber keiner war erfolgreich.)

(e) Schreiben Sie ein Programm, das unter Benutzung der Konstanten und Typen aus (a) und der Prozeduren aus (b) und (c) die Einsendungen aus der externen Einsendedatei sortiert verbucht und anschließend unter Benutzung der Prozedur aus (d) ausgibt. Die dabei angelegten Listen brauchen nicht gelöscht zu werden.

Hinweis: Sie dürfen die Lösungen der einzelnen Teilaufgaben in den darauffolgenden Teilaufgaben auch dann voraussetzen, wenn Sie sie nicht bearbeitet haben.