7. Übung – Informatik I für VIW

Fakultät Verkehrswissenschaften
Fachrichtung Verkehrsingenieurwesen

Zeitraum: 24.01.2011 bis 04.02.2011 (WS 2010/11)

Aufgabe 1 – Einfache Listenmanipulation

Gegeben ist der folgende Ausschnitt eines C-Programms:

```
struct el {int wert; struct el *next;} s1,s2,s3,s4,s5;
struct el *wurzel;
s1.wert = 1; s2.wert = 2; s3.wert = 3; s4.wert = 4; s5.wert = 5;
```

(a) Schreiben Sie eine Folge von Anweisungen, so dass `wurzel` auf eine Liste mit der Folge 4, 2, 3, 1, 5 zeigt. Verändern Sie dazu nur die Wurzel und die `next`-Pointer.

(b) Gehen Sie von der Situation aus Aufgabenteil (a) aus. Schreiben Sie eine Folge von Anweisungen, die die Liste auf die Folge 1, 5, 4, 2, 3 ändert. Verändern Sie dazu nur die Wurzel und die `next`-Pointer.

(c) Gehen Sie von der Situation aus Aufgabenteil (b) aus. Die Zahl 6 soll in die Liste hinter der Zahl 4 eingefügt werden. Schreiben Sie dazu eine Folge von Anweisungen. Achtung: Sie müssen zuerst eine neue Listenelementstruktur allokieren!

Aufgabe 2 – Einfache Listenfunktionen

(a) In einer Liste soll eine Folge von Datumsangaben (also jeweils ein Tag, Monat und Jahr) gespeichert werden. Definieren Sie eine geeignete Listenelementstruktur.

(b) Schreiben Sie eine Funktion, die zwei Pointer auf Datums-Listenelementstrukturen (wie in Aufgabenteil (a) definiert) als Argument erhält und die beiden Datumsangaben in den Strukturen, auf die die Pointer verweisen, vergleicht. Die Funktion soll 1 zurückgeben, wenn das erste Datum vor dem zweiten Datum liegt oder wenn beide Angaben gleich sind; ansonsten soll sie 0 zurückgeben.

(c) Schreiben Sie eine Funktion, die eine Wurzel auf eine Liste von Datumsangaben als Argument erhält und 1 zurückgibt, wenn die Liste aufsteigend sortiert ist, ansonsten soll sie 0 zurückgeben. Nutzen Sie dazu die Funktion aus Aufgabenteil (b).

(d – Zusatz) Schreiben Sie eine Funktion, die eine Wurzel auf eine Liste von Datumsangaben als Argument erhält und das Jahr der kleinsten Datumsangabe in der Liste zurückgibt. Nutzen Sie dazu die Funktion aus Aufgabenteil (b).

Aufgabe 3 – Strukturverändernde Listenfunktionen

Gegeben sind die beiden folgenden Strukturdefinitionen:

```
struct el {int wert; struct el *next;};
struct list {int laenge; struct el *wurzel;};
```

Die Struktur `el` steht für einen einzelnen Eintrag in einer Liste (ein Listenelement) und die Struktur `list` steht für eine Liste als Ganzes.

(a) Schreiben Sie eine Folge von Variablendeklarationen und Anweisungen, in der eine Liste mit der Folge 4, 2, 5, 6 erstellt wird und die Gesamtinformation über diese Liste in einer Variable des Typs `list` abgelegt wird.

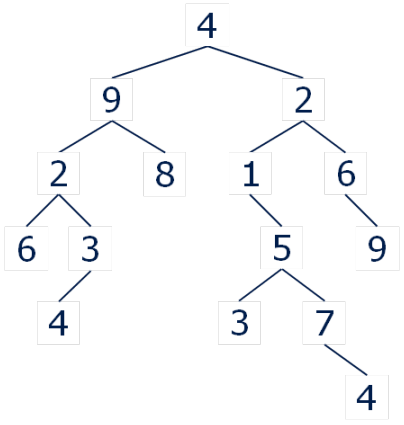
(b) Schreiben Sie eine Funktion, die einen Pointer auf eine `list`-Struktur und eine Zahl als Argumente erhält und die Zahl am Anfang der Liste einfügt. Geben Sie einen Beispielaufruf dieser Funktion für die Liste aus Aufgabenteil (a) an. Hinweis: Sie müssen zunächst eine neue Listenelementstruktur allokieren.

(c – Zusatz) Schreiben Sie eine Funktion, die einen Pointer auf eine `list`-Struktur erhält und in der entsprechenden Liste das erste Element an das Ende der Liste verschiebt. Geben Sie einen Beispielaufruf dieser Funktion für die Liste aus Aufgabenteil (a) an.

(d – Zusatz) Schreiben Sie eine Funktion, die einen Pointer auf eine `list`-Struktur erhält und eine neue Liste in umgekehrter Reihenfolge erstellt. Die Funktion soll dazu eine neue `list`-Struktur für die neue Liste erstellen und einen Pointer auf diese neue Struktur zurückgeben. Nutzen Sie dazu die Funktion aus Aufgabenteil (b).

Aufgabe 4 – Bäume und Suchbäume

(a) Lesen Sie die in dem folgenden Binärbaum abgespeicherte Zahlenfolge ab:



(b) Fügen Sie die folgenden Zahlen in der gegebenen Reihenfolge schrittweise in einen anfangs leeren Suchbaum ein: 3, 1, 10, 8, 12, 2, 9, 5, 6, 4, 7, 11. Zeichnen Sie den fertigen Suchbaum, nachdem alle Zahlen eingefügt wurden.

(c) Geben Sie eine Folge von 15 Zahlen an, so dass ein perfekt balancierter Suchbaum entsteht, wenn die Zahlen in der gegebenen Reihenfolge in einen anfangs leeren Suchbaum eingefügt werden. Gibt es eine andere Folge, die aus den gleichen Zahlen besteht und ebenfalls zu einem perfekt balancierten Baum führt?

Geben Sie außerdem eine entsprechende Folge von 15 Zahlen an, so dass ein nach links abfallender Kamm entsteht.

(d – Zusatz) Gegeben ist die folgende Struktur für einen Knoten in einem Binärbaum:

```
struct knoten {int wert; struct knoten *left, *right;};
```

Schreiben Sie eine Funktion, die einen Pointer auf den Wurzelknoten eines Binärbaumes erhält und die in dem Binärbaum gespeicherte Folge auf dem Bildschirm ausgibt. Hinweis: die Funktion müssen Sie rekursiv definieren!

Schreiben Sie eine Funktion, die einen Pointer auf den Wurzelknoten eines Binärbaumes erhält und 1 zurückgibt, wenn der Binärbaum ein Suchbaum ist; ansonsten soll die Funktion 0 zurückgeben.