Algorithmen und Datenstrukturen

Aufgabe 1 (AGS 3.2.46 b, AGS 3.2.41 ⋆)

Gegeben sei der folgende Datentyp für einfach-verkettete Listen:

```
typedef struct element *list;
struct element { int value;  list next; };
```

(a) Geben Sie eine Funktion `void append(list *lp, int n)` an, welche ein neues Element e mit Schlüsselwert `n` erzeugt, dieses hinten an eine gegebene Liste `*lp` anhängt, und den Nachfolger von e auf `NULL` setzt. Gehen Sie davon aus, dass `lp != NULL` ist.
Geben Sie Aufrufe der Funktion `append` an, sodass eine Liste mit den Elementen $4, 2, 0$ erzeugt wird.

(b) Implementieren Sie eine Funktion `sum`, welche die Werte einer solchen Liste aufsummiert.

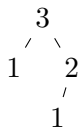
(c) Implementieren Sie eine Funktion `rmEvens`, welche aus einer Liste alle Elemente mit einer geraden Zahl entfernt. Dabei soll die bestehende Liste verändert werden.

Aufgabe 2 (3.2.36 b, 3.2.46 c)

Gegeben ist die Typdefinition für Binärbäume:

```
typedef struct node *tree;
struct node { int key; tree left, right; };
```

(a) Geben Sie eine Funktion `tree createNode(int n, tree l, tree r)` an, welche einen Binärbaum erzeugt, sodass dessen Wurzel den Schlüsselwert `n` hat und die Nachfolger der Wurzel `l` und `r` sind.

Geben Sie Aufrufe der Funktion an, um den Baum  zu erzeugen.

(b) Geben Sie eine Funktion `void insertl(tree *tp, int n)` an, welche einen Knoten mit dem Schlüsselwert `n` als am weitesten linksstehenden Knoten in den Baum `*tp` einfügt. Gehen Sie davon aus, dass `tp != NULL` ist.

(c) Ein *Blatt* ist ein Knoten, dessen linker und rechter Teilbaum jeweils leer ist. Implementieren Sie eine Funktion `int leafprod(tree t)`, welche für einen Eingabebaum `t` das Produkt der Knotenbeschriftungen der *Blätter* in `t` berechnet!

(d) Schreiben Sie eine Funktion `treeToList`, welche aus einem beliebigen Binärbaum t des oben genannten Typs eine Liste l der geraden Schlüsselwerte (`key`) von t generiert. Dabei soll l die Schlüsselwerte in folgender Reihenfolge enthalten:

- zunächst die geraden Schlüsselwerte im linken Teilbaum von t,
- dann gegebenenfalls den Schlüsselwert am Wurzelknoten, und
- zuletzt die geraden Schlüsselwerte im rechten Teilbaum von t.

Geben Sie alle dazu erforderlichen Variablendeklarationen und einen Aufruf Ihrer Funktion an. *Hinweis*: Nutzen Sie die Funktion `append` aus der vorherigen Aufgabe.

Zusatzaufgabe 1 (AGS 3.1.18 b ⋆)
Geben Sie eine Funktion `int primfac(int a[], int s)` an, die, für jedes Array `a` der Länge `s`, die Primfaktorzerlegung von `s` berechnet und in `a` speichert. Außerdem soll die Funktion die Anzahl der Faktoren in der Primfaktorzerlegung von `s` zurückgeben. Das heißt, wenn `primfac(a, s) == z` gilt, dann soll gelten:

- die ersten `z` Zahlen in `a` sind Primzahlen, und
- das Produkt der ersten `z` Zahlen in `a` ist `s`. Alle anderen Werte in `a` sind irrelevant.

Beispielsweise soll nach den Befehlen `int a[24]; int z = primfac(a, 24);` gelten: `z == 4`, und die ersten `4` Zahlen in `a` sind `2, 2, 2, 3`.

Zusatzaufgabe 2 (AGS 3.2.47 b ⋆)
Nutzen Sie die Definition des Datentype `list` aus Aufgabe 1 und geben Sie eine Funktion `void delmax(list * l)` an, welche aus einer Liste genau die Elemente entfernt, bei denen der Wert des Datenfeldes `value` am **größten** ist. Falls die Liste leer ist, soll sie nicht verändert werden. Der Speicher entfernter Elemente soll **freigegeben** werden. Falls Sie eigene Hilfsfunktionen verwenden, geben Sie diese vollständig an!