Algorithmen und Datenstrukturen

8. Übungsblatt

Zeitraum: 08. – 12. Dezember 2014

Übung 1 (AGS 3.2.33)

(a) Gegeben sei folgende Typdefinition für einfach verkettete Listen:

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

Schreiben Sie eine **iterative** Funktion `int f(list l)`, die genau dann `1` zurück gibt, wenn in der Liste alle jeweils benachbarten Elemente maximal um `1` voneinander abweichen. Ansonsten soll die Funktion `0` zurück geben.

(b) Gegeben sei die folgende Typdefinition für binäre Bäume:

```
typedef struct treeNode *tree;
typedef struct treeNode {
  int value;
  tree left, right;
} treeNode;
```

Ein *Blattknoten* ist ein Knoten, dessen linker und rechter Teilbaum jeweils leer ist. Schreiben Sie eine **rekursive** Funktion `void defol(tree *p)`, welche alle Blattknoten des übergebenen Baumes entfernt.

Übung 2 (AGS 6.1.1 ⋆)

(a) Gegeben sei die Folge: 9, 5, 4, 2, 3, 8, 1. Wenden Sie auf diese Folge den Quicksort-Algorithmus an. Dokumentieren Sie dabei die Position des Pivotelements, die Stellung der Zeiger i und j vor jedem Austauschschritt, sowie am Anfang und am Ende jedes Durchlaufs.

(b) Beurteilen Sie obige Folge bezüglich der Laufzeit von Quicksort.

Zusatzaufgabe 1 (AGS 6.1.2)
Gegeben sei die Folge: 4, 8, 2, 1, 5. Wenden Sie auf diese Folge den Quicksort-Algorithmus an!

Zusatzaufgabe 2 (AGS 3.2.34)
Gegeben sei die folgende Typdefinition für binäre Bäume:

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

(a) Der *Rang* eines Baumes t vom Typ `tree` ist die Länge des Pfades von der Wurzel von t bis zum am weitesten rechts stehenden Blatt von t. Schreiben Sie eine Funktion **`int`** `rank(tree t)`, welche den Rang des übergebenen Baums `t` berechnet! Für `t == NULL` soll `rank(t) == 0` sein.

(b) Ein Baum t heißt *Linksbaum*, falls für jeden Knoten v von t der Rang (vgl. **(a)**) des linken Teilbaums von v größer oder gleich dem Rang des rechten Teilbaums von v ist. Schreiben Sie eine Funktion **`int`** `isLeftist(tree t)`, welche für einen Baum `t` als Eingabe den Wert 1 ausgibt, falls `t` ein Linksbaum ist, ansonsten den Wert 0.

Zusatzaufgabe 3 (AGS 4.18)

```
 1  #include <stdio.h>
 2
 3  void g(int n, int *p);
 4
 5  void f(int m, int *q) {
 6    /* label1 */
 7    if (m > 0) {
 8      g(m - 1, q); /* $1 */
 9      /* label2 */
10      g(m - 2, &m); /* $2 */
11      *q = *q + m;
12    } else {
13      *q = 1;
14    }
15    /* label3 */
16  }
17  void g(int n, int *p) {
18    int x;
19    /* label4 */
20    if (n < 0) {
21      *p = 3;
22    } else {
23      f(n, &x); /* $3 */
24      *p = 2 * x;
25    }
26    /* label5 */
27  }
28
29  int main() {
30    int x;
31    /* label6 */
32    f(1, &x); /* $4 */
33    printf("%d\n", x);
34    /* label7 */
35    return 0;
36  }
```

(a) Geben Sie den Gültigkeitsbereich jedes Objektes des Programms an. Nutzen Sie dazu die Zeilennummern.

(b) Erstellen Sie ein Speicherbelegungsprotokoll für den Programmablauf.