Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

21. August 2012

Aufgaben zur Klausur **Algorithmen und Datenstrukturen in C** im SS 2012 (BInf 201, BTInf 201, BMInf 201, BWInf 201)

Zeit: 90 Minuten erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.

Sollten Unklarheiten oder Mehrdeutigkeiten bei der Aufgabenstellung auftreten, so notieren Sie bitte, wie Sie die Aufgabe interpretiert haben.

Viel Erfolg!

Diese Klausur besteht einschließlich dieses Deckblattes aus 9 Seiten.

Aufgabe 1:

Gegeben sei das folgende Programm

```
#include <stdio.h>

int x1 [] = {1,2,3};
int x2 [] = {4,2,6};
int x3 [] = {9,7,5};

int * a[] = {x3,x2,x1};
int ** pp, *p, i;

int main(void) {

  pp = a, i = * * pp;
                                        printf("%d\n",i);
  p = *(++pp), i = *(++p);
                                        printf("%d\n",i);
  i = *( *(a + 1 ) +1 );
                                        printf("%d\n",i);
  pp = a + 1, i = pp[1][2];
                                        printf("%d\n",i);
  pp = a + 1, p = pp[-1] + 1; i = p[-1];
                                        printf("%d\n",i);
  pp = a + 2, --pp, i = (*pp == a[1]);
                                        printf("%d\n",i);
  p = a[2] + 1, i = *p;
                                        printf("%d\n",i);
  pp = 1 + a, p = 2 + *pp, i = *(1 + p);
                                        printf("%d\n",i);
  return 0;
}
```

Tipp: Skizzen sind manchmal hilfreich.

Wird in diesem Programm an irgend einer Stelle eine implizite Konversion durchgeführt?

ja ☐ nein ☐

Begründung:

. . . . . .

Wie sieht die Ausgabe des Programms aus ?

1) . . . . . .

2) . . . . . .

3) . . . . . .

4) . . . . . .

5) . . . . . .

6) . . . . . .

7) . . . . . .

8) . . . . . .

Aufgabe 2:

Entwickeln Sie Makros zur Erzeugung von Bitmasken. Die Makros sollen Ausdrücke vom Typ **unsigned int** erzeugen. Sie sollen unabhängig von der Zahlendarstellung in der Maschine, 1–er oder 2–er-Komplement, sein.

1. Ein Makro `low_zeroes(n)` zum Setzen der `n` niederwertigen Bits auf 0, alle anderen Bits sollen auf 1 gesetzt werden.

   .....

   .....

   .....

2. Ein Makro
   low_ones(n) zum Setzen der `n` niederwertigen Bits auf 1, alle anderen Bits sollen auf 0 gesetzt werden.

   .....

   .....

   .....

3. Ein Makro
   mid_zeroes(width,offset), das die niederwertigen `offset` Bits auf 1 setzt, die folgenden `width` Bits auf 0, und alle übrigen wieder auf 1.

   .....

   .....

   .....

4. Ein Makro
   mid_ones(width,offset), das die niederwertigen `offset` Bits auf 0 setzt, die folgenden `width` Bits auf 1, und alle übrigen wieder auf 0.

   .....

   .....

   .....

Aufgabe 3:

Seien f_1, f_2, g_1, g_2 Funktionen vom Typ $\mathbb{N} \to \mathbb{R}$.
Weiter gelte $f_i(n) \in O(g_i(n))$ und $c_i \in \mathbb{R}$ für $i \in \{1, 2\}$.

1. Aus welcher Komplexitätsklasse ist $f(n) = f_1(n) + c_1 * n^2$

   .....

2. Aus welcher Komplexitätsklasse ist $f(n) = f_1(n) * n + c_1 * f_1(n) * n$

   .....

3. Aus welcher Komplexitätsklasse ist $f(n) = f_1(n) * (f_2(n) + c_2 * n)$

   .....

4. Aus welcher Komplexitätsklasse ist $f(n) = (f_1(n))^2 + f_1(n)$

   .....

Aufgabe 4:

Gegeben sei das folgende (unvollständige) C–Programmstück für die Implementierung von binären Suchbäumen. Alle Programmteile, die zur Lösung der Aufgabe nicht notwendig sind, sind hier weggelassen.

```
typedef int Element;

int compare(Element e1, Element e2) {
   return (e1 >= e2) − (e1 <= e2);
}

typedef struct node ∗ BinTree;
struct node {
      Element info;
      BinTree l;
      BinTree r;
};

#define isEmpty(b) (! (b))

int searchMax(Element e, BinTree t, Element ∗ max);
```

Entwickeln Sie die Routine **searchMax**. Diese Funktion soll das größte Element in dem Baum suchen, das kleiner oder gleich dem Parameter e ist. Sie soll als Funktionsresultat berechnen, ob ein solches Element existiert. Im Parameter max soll im Fall der Existenz der gesuchte Wert zurückgegeben werden.

```
int searchMax (Element e, BinTree t, Element * max)
{
  if (isEmpty (t))
      ..........
      ..........
  switch (compare (e, t->info))
    {
    case -1:
      ..........
      ..........
      ..........
      ..........
    case 0:
      ..........
      ..........
      ..........
      ..........
    case +1:
      ..........
      ..........
      ..........
      ..........
    }
}
```

Sei n die Anzahl der Elemente, die in einem Baum gespeichert sind.

1. Mit welcher Zeitkomplexität arbeitet diese Funktion im Mittel?

   .....

2. Mit welcher Zeitkomplexität arbeitet diese Funktion im schlechtesten Fall?

   .....

3. Mit welcher Zeitkomplexität würde diese Funktion arbeiten, wenn eine unsortierte verkettete Liste zur Speicherung der Elemente verwendet werden würde?

   .....

4. Mit welcher Zeitkomplexität würde diese Funktion arbeiten, wenn eine sortierte verkettete Liste zur Speicherung der Elemente verwendet werden würde?

   .....

5. Mit welcher Zeitkomplexität würde diese Funktion arbeiten, wenn eine binäre Halde zur Speicherung der Elemente verwendet werden würde?

   .....

Aufgabe 5:

Das folgende Programmstück definiert die Datentypen für die Implementierung einer Liste als einfach verkettetem Ring.

```
typedef char * Element;

typedef struct Node * Ring;
struct Node {
   Ring next;
   Element e;
};

int isEmpty(Ring r) { return r == (Ring)0; }
Element head(Ring r) { return r -> next -> e; }

extern Ring concat(Ring r1, Ring r2);
```

Entwickeln Sie den fehlenden Funktionsrumpf der *concat*–Routine zur Konkatenation zweier Ringe.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Mit welcher Zeitkomplexität arbeitet diese Routine, wenn n_1 und n_2 die Längen der beiden Listen bezeichnen.

. . . . . .