Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

28. Januar 2016

Aufgaben zur Klausur **Systemnahe Programmierung** im WS 2015/16 (B_Inf, B_TInf, B_MInf, B_WInf, B_Ecom)

Zeit: 75 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.

Nutzen Sie die Rückseiten der Klausur zur Entwicklung der Lösungen und übertragen die fertigen Lösungen in das Aufgabenblatt.

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

Tipp: Bei der Entwicklung der Lösung können kleine Skizzen manchmal hilfreich sein.

Viel Erfolg!

Diese Klausur besteht einschließlich dieses Deckblattes aus 8 Seiten.

Aufgabe 1:

Gegeben sei das folgende C–Programm zur Verarbeitung von Mengen als Bitstrings.

```
#include <stdio.h>

typedef unsigned char Set;
#define SetMax 8

void printSet(Set s) {
    unsigned int i = SetMax;
    while ( i-- != 0 ) {
        printf("%1u", (unsigned int)((s >> i) & 1));
        if (i == 4)
            printf("  ");
    }
}

#define PRINT(s) { printSet(s); printf("\n"); }

#define single(i) ( (Set)(1 << (i)) )
#define first(n) (single(n) - 1)
#define interval(n,m) (first(m+1) ^ first(n))

int main(void) {
  Set s1, s2;

  s1 = 64 + 32 + 8 + 4 + 1; PRINT(s1);
  s2 = -s1; PRINT(s2);
  s2 = ~s1 + 1; PRINT(s2);
  s2 = 2 * s1; PRINT(s2);
  s2 = s1 ^ single(3); PRINT(s2);
  s2 = s1 & 0x3c; PRINT(s2);
  s2 = s1 && (s1 - 4); PRINT(s2);
  s2 = s1 ^ (s1 & (~s1 + 1)); PRINT(s2);
  s2 = s1 & interval(2,6); PRINT(s2);
  s2 = (s1 ^ s1) & (~s1 + 1); PRINT(s2);
  s2 = s1 | 0xf0; PRINT(s2);
  s2 = 0xda >> 4; PRINT(s2);

  return 0;
}
```

Die Mengen sind in diesem Beispiel 8 Bits lang, können also die Elemente $0, 1, \ldots, 7$ enthalten. *printSet* gibt eine Menge im Binärformat aus. Die Menge, die nur die 1 enthält würde als $0000\ 0010$ ausgegeben werden. Das *PRINT* Makro gibt jeweils eine Menge pro Zeile aus. Hexadezimalzahlen werden in C mit dem Präfix $0x$ gekennzeichnet und erlauben die Ziffern $0, 1, \ldots, 9, a, b, c, d, e, f$.

Welche 12 Ausgabezeilen erzeugt dieses Programm unter der Annahme, dass die Maschine mit 2er-Komplement-Zahlendarstellung arbeitet?

Vorsicht: Berechnen Sie den Wert für die Variable $s1$ mit größter Sorgfalt.

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

8) .....

9) .....

10) .....

11) .....

12) .....

Aufgabe 2:

Gegeben sei das folgende C–Programm:

```
#include <stdio.h>

#define minm(x,y) ((x) < (y) ? (x) : (y))

long minf(long x, long y) { return x < y ? x : y; }

unsigned ausgewertet = 0;

#define e(x) ( ++ausgewertet, (x) )


int main(void) {
    long f[] = { 88, −6, −2, 42, 13 };
    long r0;

    double g[] = { 3.14, 0.25, 2.11, 6.66 };
    double r1;

    ausgewertet = 0;
    r1 = minm( e(g[2]), e(g[3]) );

    printf(" r1 = %4.2f, ausgewertet = %u\n", r1, ausgewertet);

    ausgewertet = 0;
    r1 = minf( e(g[2]), e(g[3]) );

    printf(" r1 = %4.2f, ausgewertet = %u\n", r1, ausgewertet);

    ausgewertet = 0;
    r0 = minm( minm( e(f[4]), e(f[3]) ),
               e(f[0]) );

    printf(" r0 = %ld, ausgewertet = %u\n", r0, ausgewertet);

    ausgewertet = 0;
    r0 = minf( minf( e(f[3]), e(f[2]) ),
               e(f[4]) );

    printf(" r0 = %ld, ausgewertet = %u\n", r0, ausgewertet);

    return 0;
}
```

Welche vier Ausgabezeilen erzeugt dieses Programm:

1) .....

2) .....

3) .....

4) .....

Aufgabe 3:

Gegeben sei das folgende Programm:

```
#include <stdio.h>

char * tab [] = { "Falle" , "Maus" , "Kaefig" , "Gericht" };

char * * ptab [] = { tab + 3, tab + 2, tab + 1, tab };

char * * * ppp = ptab;

int main ( int argc , char * argv [] )
{
   printf( "%s\n" , * (* (ppp + 2) - 1) + 1 );
   printf( "%s\n" , ppp [2] [0] + 3 );
   printf( "%s\n" , * (* ++ppp + 1) + 2 );
   printf( "%s\n" , * * ppp + 4 );

   return 0;
}
```

Welche Ausgabezeilen liefert dieses Programm:

1) .....

2) .....

3) .....

4) .....

Tipp: Machen Sie auf einer Rückseite der Klausur zur Veranschaulichung eine Skizze von den im Programm initialisierten Variablen.

Wie viel Speicher wird von den Variablen tab, ptab, ppp und den in den Initialisierungen vorkommenden Konstanten benötigt? Geben Sie hierfür einen Ausdruck mit dem **sizeof**-Operator an (z.B. 5 * sizeof(char) + 2 * sizeof(char *) + ...)

.....

.....

Aufgabe 4:

Gegeben seien die C Makro-, Typ-, Variablen- und Funktionsdeklarationen und -definitionen:

```
typedef double Element;

typedef Element *Row;
typedef Row *Rows;

struct Descr
{
  Rows rows;
  Element *elems;
  int width;
  int height;
};

#define trunc(x) ((int)(x))
#define at(a,i,j) ((a)->rows[i][j])

typedef struct Descr * Matrix;

typedef Matrix (* MatrixConstr)(int, int);
typedef Matrix (* UnaryMatrixOp)(Matrix);
typedef Matrix (* BinaryMatrixOp)(Matrix, Matrix);

extern Matrix newMatrix (int w, int h);
extern Matrix zeroMatrix (int w, int h);
extern Matrix unitMatrix (int w, int h);

extern Matrix addMatrix (Matrix m1, Matrix m2);
extern Matrix transposeMatrix (Matrix m);

Matrix m1;
MatrixConstr c1;
```

Bestimmen Sie für die folgenden Ausdrücke den Typ gemäß ANSI-C. Sollten fehlerhafte oder logisch nicht sinnvolle Ausdrücke vorkommen, so kennzeichnen Sie diese mit dem Wort FEHLER. Nutzen Sie für die Typbestimmung, soweit es möglich ist, die Namen der in dem Programmstück deklarierten Typen.

m1->elems .....

*((*m1).elems) .....

m1->rows[1][2] .....

m1->rows[m1->width] .....

m1->rows[1,2] .....

at(m1,0,0) .....

at(m1,trunc(at(m1,1,1)),+1) .....

zeroMatrix .....

c1(1,1) .....

at(c1(1,1),1,1) .....

c1 = unitMatrix .....

c1 = transposeMatrix .....

(*addMatrix(m1,m1)).rows + 1 .....
