Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

3. März 2016

Aufgaben zur 2. 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.

Vorsicht: Lesen gefährdet die Dummheit!

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(i,s) { printf("%2u)  ",i); printSet(s); printf("\n"); }

#define anInterval(n,m) (theFirst(m+1) ^ theFirst(n))
#define theFirst(n) (oneElement(n) − 1)
#define oneElement(i) ( (Set)(1 << (i)) )

int main(void) {
    Set s1;
    PRINT(1, oneElement(3) );
    PRINT(2, oneElement(SetMax−1) );
    PRINT(3, oneElement(SetMax) );

    PRINT(4, (Set)5 );
    PRINT(5, (Set)55 );

    PRINT(6, anInterval(1,SetMax−3) );
    PRINT(7, anInterval(2,2) );
    PRINT(8, anInterval(6,5) );

    PRINT(9, 29 & 28 );
    PRINT(10, 29 && 28 );

    s1 = ∼anInterval(1,4) | anInterval(3,6); PRINT(11, s1);
    s1 = anInterval(1,6) ^ ((128 − 1) ∗ 4); PRINT(12, s1);

    s1 = 64 + 24;
    s1 = s1 ^ (s1 & (∼s1 + 1)); PRINT(13, s1);
    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 00000010 ausgegeben werden. Das *PRINT* Makro gibt jeweils eine Menge pro Zeile aus und numeriert die Zeilen durch.

Welche 13 Ausgabezeilen erzeugt dieses Programm?

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Aufgabe 2:

Gegeben sei das folgende C–Programm:

```
#include <stdio.h>

#define maxm(x,y) ((x) > (y) ? (x) : (y))

long maxf(long x, long y) { return x > y ? x : y; }

unsigned cnt = 0;

#define eval(x) ( ++cnt, (x) )


int main(void) {
    long f[] = { 0, +2, −4, +6, −8 };
    long res;

    double g[] = { 3.1, 1.5, 4.9 };
    double r;

    cnt = 0;
    res = maxm( maxm( eval(f[1]), eval(f[2]) ),
                eval(f[3]) );

    printf(" res = %ld, cnt = %u\n", res, cnt);

    cnt = 0;
    res = maxf( maxf( eval(f[1]), eval(f[2]) ),
                eval(f[3]) );

    printf(" res = %ld, cnt = %u\n", res, cnt);

    cnt = 0;
    r = maxm( eval(g[1]), eval(g[2]) );

    printf(" r = %3.1f, cnt = %u\n", r, cnt);

    cnt = 0;
    r = maxf( eval(g[1]), eval(g[2]) );

    printf(" r = %3.1f, cnt = %u\n", r, cnt);

    return 0;
}
```

Welche vier Ausgabezeilen erzeugt dieses Programm:

1) .....

2) .....

3) .....

4) .....

Aufgabe 3:

Gegeben sei das folgende Programm:

```
#include <stdio.h>

char * tab [] = { "Hasenfuss" , "Opernball" , "Esstisch" , "Memme",
"Kanister" };

char * * ptab [] = { tab + 4, tab + 3, tab + 2, tab + 1, tab };

char * * * ppp = ptab;

int main ( int argc , char * argv [] ) {
   printf( "%s\n" , * (* (ppp + 2)) + 3 );
   printf( "%s\n" , * (* (ppp + 3) − 1) + 5 );
   printf( "%s\n" , ppp [3] [0] + 5 );
   printf( "%s\n" , * (* ++ppp) + 3 );
   printf( "%s\n" , * (* −−ppp) + 3 );
   return 0;}
```

Welche Ausgabezeilen liefert dieses Programm:

1) .....

2) .....

3) .....

4) .....

5) .....

Wie viel Speicher wird von den Variablen tab, ptab und 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 folgenden Variablen und Funktionen:

```
typedef int (*Fct1)(int);
typedef int (*Fct2)(int, int);
typedef int (*Fct3)(double);
int x;
long int s;
float f;
char *p1;
int *p2;
void *p3;
Fct1 fp1, fp2;
Fct3 fp3;
int (*pf)(int);
int (*pf2)(double);
int f1(int x1);
int f2(int x1,int x2);
int f3(double x1);
```

Bestimmen Sie für die folgenden Ausdrücke den Typ gemäß ANSI-C. Kennzeichnen Sie die Ausdrücke, die fehlerhaft sind oder implizite Konversionen enthalten. mit dem Wort FEHLER

$\sim$s .....

p3[x] .....

p1 ? x : f .....

$\sim$p2 .....

p1 & p1 .....

pf=f1(x) .....

s || x .....

fp1(25) .....

pf=f1 .....

fp1==f1 .....

f=f3(f) .....

f1==f2 .....

$*$p3 .....

p2[x] .....

p1 && p1 .....

fp1==fp2 .....

pf2=pf .....