Fachhochschule Wedel
Prof. Dr. Uwe Schmidt

9. August 2002

Aufgaben zur Klausur **C** im SS 2002 (PI h742)

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.

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 6 Seiten

---

Aufgabe 1:

Gegeben seien die folgenden Variablen:

```
int x;
unsigned int u;
long int s;
float f;
char *p1;
long int *p2;
void *p3, *p4;
```

Bestimmen Sie für die folgenden Ausdrücke den Typ gemäß ANSI-C. Vorsicht: Es kommen fehlerhafte und logisch falsche Ausdrücke von. Kennzeichnen Sie diese mit dem Wort **FEHLER**

p2[*p2] .....

s || x || p1 .....

s | x | p1 .....

p1 ? x : f .....

++f, x-- .....

p1 == p3 .....

p3 == p4 .....

x = p3 == p4 .....

x == p3 == p4 .....

1 + ~p2 .....

p1 && p3 .....

x += f .....

!!! p3 .....

!!! s .....

~~~s .....

0x23 + p1 + 0x23 .....
~~~

Aufgabe 2:

Gegeben sei das folgende Programm:

```
#include <stdio.h>

char * tab [] = { "Dreyfuss" , "Kimball" , "dwelt" , "Kodachrome", "McAllister" };

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

char * * * ppp = ptab;

int main ( int argc , char * argv [] )
{
  printf( "%s\n" , * (* (ppp + 3) - 1) + 4 );
  printf( "%s\n" , ppp [3] [0] + 3 );
  printf( "%s\n" , * (* (ppp + 2)) + 1 );
  printf( "%s\n" , * (* ++ppp) + 8 );
  printf( "%s\n" , * (* --ppp) + 5 );

  return 0;
}
```

Welche Ausgabezeilen liefert dieses Programm:

1) .....

2) .....

3) .....

4) .....

5) .....

Wieviel 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.

.....

.....

Aufgabe 3:

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));
}

static unsigned int linecnt = 0;

#define PRINT(s) { printf("%2u) ", ++linecnt); 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( oneElement(1) );
    PRINT( oneElement(SetMax-1) );
    PRINT( oneElement(SetMax) );

    PRINT( (Set)3 );
    PRINT( (Set)33 );

    PRINT( anInterval(1,SetMax-3) );
    PRINT( anInterval(2,2) );
    PRINT( anInterval(6,5) );

    PRINT( 29 & 28 );
    PRINT( 29 && 28 );

    s1 = ~anInterval(1,4) | anInterval(3,6); PRINT(s1);
    s1 = anInterval(1,6) ^ ((128 - 1) * 4); PRINT(s1);

    s1 = 32 + 24;
    s1 = s1 ^ (s1 & (~s1 + 1)); PRINT(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?

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

8) .....

9) .....

10) .....

11) .....

12) .....

13) .....

Physikalisch–Technische Lehranstalt Wedel
Prof. Dr. Uwe Schmidt

9. August 2002

Aufgaben zur Klausur **C** im SS 2002 (IA 302)

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.

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 6 Seiten

---

Aufgabe 1:

Gegeben seien die folgenden Variablen:

```
int x;
unsigned int u;
long int s;
float f;
char *p1;
long int *p2;
void *p3, *p4;
```

Bestimmen Sie für die folgenden Ausdrücke den Typ gemäß ANSI-C. Vorsicht: Es kommen fehlerhafte und logisch falsche Ausdrücke von. Kennzeichnen Sie diese mit dem Wort **FEHLER**

p2[*p2] .....

s || x || p1 .....

s | x | p1 .....

p1 ? x : f .....

++f, x-- .....

p1 == p3 .....

p3 == p4 .....

x = p3 == p4 .....

x == p3 == p4 .....

1 + ~p2 .....

p1 && p3 .....

x += f .....

!!! p3 .....

!!! s .....

~~~s .....

0x23 + p1 + 0x23 .....
~~~

Aufgabe 2:

Gegeben sei das folgende Programm:

```
#include <stdio.h>

char * tab [] = { "Dreyfuss" , "Kimball" , "dwelt" , "Kodachrome", "McAllister" };

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

char * * * ppp = ptab;

int main ( int argc , char * argv [] )
{
  printf( "%s\n" , * (* (ppp + 3) - 1) + 4 );
  printf( "%s\n" , ppp [3] [0] + 3 );
  printf( "%s\n" , * (* (ppp + 2)) + 1 );
  printf( "%s\n" , * (* ++ppp) + 8 );
  printf( "%s\n" , * (* --ppp) + 5 );

  return 0;
}
```

Welche Ausgabezeilen liefert dieses Programm:

1) .....

2) .....

3) .....

4) .....

5) .....

Wieviel 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.

.....

.....

Aufgabe 3:

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));
}

static unsigned int linecnt = 0;

#define PRINT(s) { printf("%2u) ", ++linecnt); 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( oneElement(1) );
    PRINT( oneElement(SetMax-1) );
    PRINT( oneElement(SetMax) );

    PRINT( (Set)3 );
    PRINT( (Set)33 );

    PRINT( anInterval(1,SetMax-3) );
    PRINT( anInterval(2,2) );
    PRINT( anInterval(6,5) );

    PRINT( 29 & 28 );
    PRINT( 29 && 28 );

    s1 = ~anInterval(1,4) | anInterval(3,6); PRINT(s1);
    s1 = anInterval(1,6) ^ ((128 - 1) * 4); PRINT(s1);

    s1 = 32 + 24;
    s1 = s1 ^ (s1 & (~s1 + 1)); PRINT(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?

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

8) .....

9) .....

10) .....

11) .....

12) .....

13) .....