

---

Aufgaben zur Klausur C im SS 99 (IA 302)

Zeit: 60 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 7 Seiten

---

### Aufgabe 1:

Gegeben seien die folgenden Variablen und Funktionen:

```
int x;  
long int s;  
float f;  
char *p1;  
int *p2;  
void *p3;  
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. Vorsicht: Es kommen fehlerhafte Ausdrücke von. Kennzeichnen Sie diese entsprechend

- p2[x] .....
- p3[x] .....
- p1 ? f1 : pf .....
- ~p2 .....
- p1 && p1 .....
- ~s .....
- s || x .....
- (\*pf)(5) .....
- pf=f1 .....
- pf=f1(x) .....
- pf2=pf .....
- pf==f1 .....
- pf==pf2 .....
- f=f3(f) .....
- f1==f2 .....
- \*p3 .....

## Aufgabe 2:

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));
}

#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;

    PRINT( (Set)0 );
    PRINT( (Set)-1 );
    PRINT( single(SetMax / 2 + 1) );

    PRINT( interval(0,1) );
    PRINT( interval(2,5) );
    PRINT( interval(0,SetMax) );

    PRINT( 3 | 8 );
    PRINT( 3 || 8 );
    PRINT( first(SetMax / 2 + 1) );

    s1 = interval(1,5) & ~interval(2,6); PRINT(s1);
    s1 = interval(1,5) ^ interval(2,6); PRINT(s1);

    s1 = 4 + 16 + 32;
    s1 ^= s1 & (~s1 + 1); PRINT(s1);

    return 0;
}
```

Die Mengen sind in diesem Beispiel 8 Bits lang, können also die Elemente  $0, 1, \dots, 7$  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.

Welche 12 Ausgabezeilen erzeugt dieses Programm?

- 1) .....
- 2) .....
- 3) .....
- 4) .....
- 5) .....
- 6) .....
- 7) .....
- 8) .....
- 9) .....
- 10) .....
- 11) .....
- 12) .....

### Aufgabe 3:

Gegeben sei das folgende C-Programm

```
int x1 [] = {11,12,13};
int x2 [] = {40,20,60};
int x3 [] = {99,77,55};

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

int main(int argc, char * argv[]) {

    /* 1. */ pp = a;
    /* 2. */ pp = a-1;

    /* 3. */ p = *p;
    /* 4. */ p = *(a + 1);
    /* 5. */ p = *a + 1;
    /* 6. */ p = *(a++);
    /* 7. */ pp = a, p = *(--pp);
    /* 8. */ pp = a-1, p = *(++pp);
    /* 9. */ pp = a, p = *(pp++) -1;

    /* 10. */ c = **a;
    /* 11. */ c = *a;
    /* 12. */ c = a;

    /* 13. */ c = a[2][3];
    /* 14. */ c = a[3][3];
    /* 15. */ c = a[3,3];

    return 0;
}
```

1. Ist  $a$  ein Feld von Zeigern ?

ja  nein  weiß nicht

2. Ist  $a$  ein Zeiger auf ein Feld ?

ja  nein  weiß nicht

3. Wieviel Platz benötigt  $a$  (ein Ausdruck mit `sizeof` Operator)?

.....

4. Wieviel Platz wird im Datensegment insgesamt für die Variablen  $x_1, x_2, x_3, a, pp, p$  und  $c$  benötigt (ein Ausdruck mit `sizeof` Operator)?

.....  
.....  
.....

Überprüfen sie die Anweisungen im Hauptprogramm auf korrekte Kompilierbarkeit, auf implizite Konversionen und auf nicht definierte Wirkung zur Laufzeit (Semantik undefiniert  $\Rightarrow$  Ergebnisse Zufall).

- |     |   |  |  |
|-----|---|--|--|
| 1.  | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 2.  | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 3.  | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 4.  | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 5.  | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 6.  | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 7.  | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 8.  | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 9.  | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 10. | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 11. | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 12. | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 13. | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 14. | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |
| 15. | Compilerfehler <input type="checkbox"/> | impl.Konversion <input type="checkbox"/> | Wirkung undefiniert <input type="checkbox"/> |