

Aufgaben zur Klausur C im WS 2008/09 (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 8 Seiten.

Aufgabe 1:

Gegeben sei das folgende C-Programmstück für die Verarbeitung von binären Bäumen:

```
#include <stdlib.h>
#include <string.h>

typedef char * String;

typedef struct node * BinTree;
struct node {
    String info;
    BinTree l;
    BinTree r;
};

extern BinTree mkEmpty(void);
extern BinTree mkOne(String s);
extern BinTree insert(String s, BinTree t);

extern int isEmpty(BinTree t);
extern int isIn(String s, BinTree t);

extern int invBinTree(BinTree t);
```

Implementieren Sie die fehlende Testroutine *invBinTree*. Diese überprüft, ob ein binärer Suchbaum konsistent aufgebaut ist, das heißt, für alle Knoten in dem Baum gilt, dass alle Elemente im linken Teilbaum kleiner als das Element im Knoten sind und alle Elemente im rechten Teilbaum größer. Versuchen Sie, überflüssige Vergleiche zu vermeiden. Zerlegen Sie, wenn nötig, die Aufgabe in mehrere Funktionen.

Nutzen Sie die in dem Programmstück vorgegebenen Datentypen und Funktionsdeklarationen. (Lösung auf der nächsten Seite).

Die *invBinTree* Routine:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Aufgabe 2:

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

```
typedef double Random;
```

```
typedef Random (*F) (void);
```

```
typedef struct x *R;
```

```
struct x  
{  
    unsigned int n;  
    Random t;  
    R l[2];  
    R *p;  
    F getRandom;  
    int x;  
    unsigned char c[5];  
};
```

```
R x1;
```

```
long int i;
```

```
double rf1 (void);
```

```
double rf2 (double x);
```

und die folgenden C-Ausdrücke

1. $\&(x1 \rightarrow p)$
2. $x1 \rightarrow \text{getRandom} == \text{rf1}$
3. $*(x1 \rightarrow l[1])$
4. $x1 \rightarrow l$
5. $*(x1 \rightarrow l)$
6. $(x1 \rightarrow \text{getRandom})() = \text{rf2}(1.0)$
7. $x1 \rightarrow c$
8. $*((*x1).c)$
9. $*(x1 \rightarrow c) \& i$
10. $i ? x1 \rightarrow x : x1 \rightarrow n$
11. $x1 \&\& i$
12. $(x1 \rightarrow \text{getRandom}) = \text{rf2}$

1. Welche Ausdrücke sind fehlerhafte C-Ausdrücke oder enthalten logische Fehler? (Diese Ausdrücke sind in den folgenden Fragen nicht mehr zu berücksichtigen).

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

2. Welche Ausdrücke besitzen einen vorzeichenlosen ganzzahligen Typ?

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

3. Welche Ausdrücke besitzen einen *struct x*-Typ?

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

4. Welche Ausdrücke besitzen einen Typ *Zeiger auf ...*?

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

5. Welche Ausdrücke besitzen einen Typ *Zeiger auf Zeiger auf ...*?

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

6. Welche Ausdrücke besitzen einen Funktionszeiger als Typ?

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

7. Welche Ausdrücke werden bei beliebiger Variablenbelegung immer zu 0 oder 1 ausgewertet?

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

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));
        if (i == 4)
            printf(" ");
    }
}

static unsigned int linecnt = 0;
#define PRINT(s) { printf("%2u)  ", ++linecnt); 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 = 0xad; PRINT(s1);
    s2 = 1-s1; PRINT(s2);
    s2 = -s1; PRINT(s2);
    s2 = ~s1 + 1; PRINT(s2);
    s2 = s1 & 0x3c; PRINT(s2);
    s2 = s1 | 0xf0; PRINT(s2);
    s2 = s1 && (s1 - 4); PRINT(s2);
    s2 = (s1 ^ s1) & (~s1 + 1); PRINT(s2);
    s2 = s1 ^ (s1 & (~s1 + 1)); PRINT(s2);
    s2 = s1 & interval(2,6); PRINT(s2);
    s2 = s1 ^ single(3); PRINT(s2);

    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 0000 0010 ausgegeben werden. Das *PRINT* Makro gibt jeweils eine Menge pro Zeile aus.

Welche 11 Ausgabezeilen erzeugt dieses Programm unter der Annahme, dass die Maschine mit 2er-Komplement Zahlendarstellung arbeitet?

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

Aufgabe 4:

Gegeben sei das folgende Programm:

```
#include <stdio.h>

char * tab [] = { "Haschisch", "Unterleib", "Geschenk", "Blei", "Laster" };

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 ) + 6 );
    printf( "%s\n" , ppp [3] [0] + 6 );
    printf( "%s\n" , * ( * ( ppp + 2 ) ) + 5 );
    printf( "%s\n" , * ( * ++ppp ) + 1 );
    printf( "%s\n" , * ( * --ppp ) + 2 );

    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.

.....
.....