Lösung zur
Klausur Systemnahe Software I

WS 96/97

21. Januar 1997

Aufgabe		Punkte	Bewertung
1	a)	6	
	b)	6	
2		10	
3	a)	4	
	b)	4	
	c)	2	
	d)	4	
4		10	
5		10	
6	a)	6	
	b)	4	
	c)	4	
	d)	4	
7		6	
Summe		**80**	

Hinweis: Bitte benützen Sie den jeweils vorgesehenen Platz. Sollte dieser nicht ausreichen, verwenden Sie bitte die Rückseite und machen Sie eine eindeutige Markierung!

Aufgabe 1

(*a*) 6 Punkte

Gegeben sei folgende Vereinbarung: *long int a[N][M];*
Welche der folgenden Ausdrücke sind äquivalent zu a[3][2]?

- ☒ *(a[3]+2) /* korrekt */
- ☐ (*(a+2))[3] /* Reihenfolge! */
- ☒ *((*(a+3))+2) /* korrekt */
- ☐ **(a+3)+2 /* Vorrang! */
- ☐ *(&a[0][0]+3+2) /* Vorrang! */

Bitte die richtigen Antworten ankreuzen. Falsche Antworten führen natürlich zu Punktabzug!

(*b*) 6 Punkte

In einem zu erstellenden C-Programm benötigen Sie eine Datenstruktur, die folgende Teilinformationen enthalten soll:

- MatrNr als ganze Zahl;
- Name mit maximal 20 Buchstaben
- Datum mit den drei Komponenten Tag (ganze Zahl), Monat (ganze Zahl) und Jahr (ganze Zahl)

Geben Sie die Vereinbarung einer C-Variablen an, die ein Objekt wie oben beschrieben aufnehmen kann.

```
struct {
        int MatrNr;
        char Name[21];
        struct {
                int Tag;
                int Monat;
                int Jahr;
        } Datum;
} student;
```

Aufgabe 2

10 Punkte

Schreiben Sie eine Funktion *pack*, die als Parameter 4 Variable vom Typ *char* erhält und diese in eine *unsigned long int* verpackt zurückliefert.

```
unsigned long int pack(char a, char b, char c, char d)
{
        return a << 24 | b << 16 | c << 8 | d;
}
```

Diese Lösung hat allerdings einen Schönheitsfehler: Die vier Parameter müßten vom Typ *unsigned char* sein, damit negative Werte keine Probleme machen (sie würden bei der Expansion mit Einsen aufgefüllt). Korrekter wäre daher eine der folgenden Varianten:

```
unsigned long int pack1(char a, char b, char c, char d)
{
        unsigned long int result;

        result = a & 0xff;
        result <<= 8;
        result |= b & 0xff;
        result <<= 8;
        result |= c & 0xff;
        result <<= 8;
        result |= d & 0xff;

        return result;
}

unsigned long int pack2(char a, char b, char c, char d)
{
        char buf[4];

        buf[0] = a;
        buf[1] = b;
        buf[2] = c;
        buf[3] = d;

        return * (unsigned long int *) buf;
}

unsigned long int pack3(char a, char b, char c, char d)
{
        return  (unsigned long) (unsigned char) a << 24 |
                (unsigned long) (unsigned char) b << 16 |
                (unsigned long) (unsigned char) c << 8 |
                (unsigned long) (unsigned char) d;
}
```


Aufgabe 3

(*a*) 4 Punkte

Schreiben Sie ein Makro *TO_LOWER(c)*, das einen Großbuchstaben in einen Kleinbuchstaben verwandelt. Eine Prüfung, ob tatsächlich ein Großbuchstabe vorliegt, braucht nicht zu erfolgen!

```
#define TO_LOWER(c)     ((c)+'a'-'A')
```

oder:

```
#include                <ctype.h>
#define TO_LOWER(c)     tolower(c)
```

(*b*) 4 Punkte

Schreiben Sie ein Makro *MUL_BY_TWO(x)*, das sein numerisches Argument verdoppelt.

```
#define MUL_BY_TWO(x)       ((x)+(x))
```

oder:

```
#define MUL_BY_TWO(x)       ((x)*2)
```

Für die folgenden beiden Teilaufgaben seinen seien folgende Makro-Vereinbarung und Makro-Aufrufe gegeben:

```
#define min(x,y) ((x) >= (y)?x:y)

1) min(1,a)

2) min(1,min(2,3))
```

(*c*) 2 Punkte

Welchen Ausdruck ergibt die Substitution bei 1)?

```
(( 1 ) >= ( a )? 1 : a )
```

(*d*) 4 Punkte

Welchen Ausdruck ergibt die Substitution bei 2) ?

```
((1) >= ( ((2) >= (3)? 2 : 3 ) )? 1 : ((2) >= (3)? 2 : 3 ) )
          ^^^^^^^^^^^^^^^^^^^^              ^^^^^^^^^^^^^^^^^^^^
```

Aufgabe 4

10 Punkte

Gegeben ist folgendes C-Programm:

```
/* obscure.c */

int obscure(int x, int * y) {
      x++; (*y)++;
      return (x + *y);
}

main() {
     int x = 6;
     int y = 8;

     if ( (obscure(x, &y) < obscure(y, &x)) ||
          (obscure(x, &y) > obscure(y, &x)) ) {
               x = obscure(x, &y);
          y = obscure(y, &x);
     }
     else {
               x = obscure(y, &x);
          y = obscure(x, &y);
     }

     printf("x = %d\ny = %d\n", x,y);
}
```

Geben Sie an, was dieses Programm als Ausgabe liefert!

	x	y	Return-Wert
	6	8	
`if ( (obscure(x, &y)`	6	9	16
`< obscure(y, &x))` `|| ... ) {`	7	9	17
`x = obscure(x, &y);`	7 18	10	18 <–
`y = obscure(y, &x);`	19	10 30	30 <–
`} else ...` `printf(...);`	19	30	

Antwort:

```
x = 19
y = 30
```


Aufgabe 5

10 Punkte

Gegeben sei folgendes Programm, bei dem Sie die Funktion *add()* ausformulieren sollen; diese Funktion soll immer am Anfang einer Liste ein neues Element einfügen.

```
#include <stdio.h>

extern char * strcpy();

typedef struct node * ListPtr;

struct node {
     char * name;
     ListPtr next;
};

ListPtr createList() {
     return NULL;
};

ListPtr add(   ListPtr anker,      /*Zeiger auf erstes Element*/
               char * s  /* einzufuegendes Element */
        ) {
/* Ihre Aufgabe: */









}

main() {
     ListPtr list;
     char buffer[80];
     list = createList();
     gets(buffer);
     list = add(list,buffer);
     gets(buffer);
     list = add(list,buffer);
}
```

Aufgabe 6

(*a*) 6 Punkte

Geben Sie den Funktionskopf für folgende Spezifikation an:

Eine Funktion namens *attention*, die als Parameter eine Funktion, die *int* liefert, erhält und einen Zeiger auf eine Funktion, die *int* liefert, zurückgibt!

Beschreiben Sie die im folgenden deklarierten Objekte:

(*b*) 4 Punkte

```
int *(*ha)();
```

(*c*) 4 Punkte

```
int (*hi[])()
```

(*d*) 4 Punkte

```
int (*ho())[];
```


Aufgabe 7

6 Punkte

Gegeben ist folgendes Programm:

```
#define FALSE 0
#define TRUE 1

void bubble_sort(
               int list[],
               int list_size

          ) {
     int j,k, temp, sorted = FALSE;

     while (!sorted) {
          sorted = TRUE;
          for(j=0; j<list_size-1;j++)

               if (list[j] > list[j+1]) {

                    temp = list[j];
                    list[j] = list[j+1];
                    list[j+1] = temp;
                    sorted = FALSE;
               }
     }
}

int cmp_asc(int a,int b) {
     /* compare ascending */
     return (a>b);
}

int cmp_desc(int a,int b) {
     /* compare descending */
     return (a<b);
}

main() {
     int vektor[] = { 4,6,2,8,3,6,10,1 };
     int length = 8, i;
     bubble_sort(vektor,length          );
     for(i=0; i<length;i++)
          printf("%d ", vektor[i]);
     printf("\n");
}
```

Modifizieren Sie dieses Programm so, daß die Prozedur *bubble_sort* sowohl absteigend als auch aufsteigend (oder, falls andere Vergleichsfunktion gegeben sind, nach diesen) sortieren kann. Von *main()* aus soll sie für aufsteigende Sortierung aufgerufen werden.