FACHHOCHSCHULE WEDEL
Prof. Dr. Uwe Schmidt

8. Februar 1996

Aufgaben zur Klausur **C** im WS 95/96 (II21,II63)

Zeit: 120 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.

Viel Erfolg !

Diese Klausur besteht einschließlich dieses Deckblattes aus 10 Seiten

---

Aufgabe 1:

Gegeben sei das folgende C–Programm. Hinweis: der $\ll$ Operator hat geringere Priorität als + und -.

```
#include <stdio.h>
#include <math.h>

#define mal2(x) x + x
#define hoch2(x) x * x
#define max(x,y) x > y ? x : y
#define wurzel(x) (++cnt, sqrt(x))

int cnt = 0;

int main(void)
{
    int i = 3, j = 2, r;
    double f = 3.5;

    r = mal2(j << j);
    printf(".1) r = %d\n",r);

    r = hoch2(i + j);
    printf(".2) r = %d\n",r);

    r = mal2(i) * 3;
    printf(".3) r = %d\n",r);

    r = hoch2(mal2(i));
    printf(".4) r = %d\n",r);

    r = mal2(f);
    printf(".5) r = %d\n",r);

    r = max(mal2(i+1),hoch2(i));
    printf(".6) r = %d\n",r);

    r = max(wurzel(i+1),wurzel(j));
    printf(".7) r = %d, cnt = %d\n",r,cnt);

    return 0;
}
```

Welche 7 Ausgabezeilen erzeugt dieses Programm

.....

.....

.....

.....

.....

.....

.....

Ändern Sie das Makro *hoch2* so ab, daß es für `.2)` den gewünschten Effekt hat.

.....

Ändern Sie das Makro *mal2* so ab, daß es sich immer wie eine Funktion verhält

.....

Warum kann das Makro *max* nicht so verändert werden, das es sich immer, bei jedem Gebrauch, wie eine gleichwertige Funktion verhält?

.....

Warum kann der Gebrauch von Makros in der Form von *max* die Programmausführung verlangsamen (im Vergleich zu Funktionen).

.....

Aufgabe 2:

Verarbeitung von Mengen mit Hilfe binärer Bäume.

Ein binärer Baum ist eine Datenstruktur, die entweder *leer* ist oder aus einem Knoten besteht. Dieser Knoten enthält drei Komponenten: eine Komponente enthält einen Wert des Elementbereiches, die beiden anderen Komponenten enthalten wieder binäre Bäume, wobei folgende Bedingung gelten muß:

Alle Werte des einen (linken) binären Teilbaums müssen kleiner als der Wert im Knoten sein, alle Elemente des anderen (rechten) binären Teilbaums müssen größer als der Wert im Knoten sein.

Entwickeln Sie einen C–Headerfile `menge.h`, in dem folgende Objekte deklariert und definiert werden:

1. eine Typdefinition eines Typs `Element` für lange vorzeichenlose Zahlen

   .....

2. eine Typdefinition eines Typs `Menge` für einen binären Baum zur Verarbeitung von beliebigen Mengen von Werten vom Typ `Element`.

   .....

   .....

   .....

   .....

   .....

3. eine Makrodefinition `empty` für den Wert, der die leere Menge repräsentiert. Die leere Menge soll durch einen 0-Zeiger repräsentiert werden.

   .....

4. einen Prototyp für eine Funktion `insert` mit dem Resultattyp `Menge` und den Parametern `x` vom Typ `Element` und `y` vom Typ `Menge`

   .....

5. einen Prototyp für eine Funktion `is_in`, die als Resultat 0 oder 1 liefert und die gleichen Parameter hat wie `insert`

   .....

Aufgabe 3:

Gegeben sei das folgende C–Programm:

```
#include <stdio.h>

int h(int i) {
  printf("h");
  return i < 10 ? i : h( (i % 10) + (i / 10));
}

int g(int i) {
  printf("g");
  return ( (i % 10) == 7) || (i ≥ 10 && g(i / 10));
}

int f(int i) {
  printf("f");
  return (i % 7 == 0) || g(i) || (h(i) == 7);
}

int main() {
  printf(" %d\n",f(77));
  printf(" %d\n",f(61));
  printf(" %d\n",f(99));
  printf(" %d\n",f(7002));

  return 0;
}
```

Welche vier Ausgabezeilen erzeugt dieses Programm:

1) .....

2) .....

3) .....

4) .....

Aufgabe 4:

Gegeben seien die folgenden C Datentypen und Variablen

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

typedef struct Node * NodePtr;

struct Node {
    NodePtr next;
    NodePtr prev;
    char * value;
};

struct ListDescr {
    NodePtr first;
    NodePtr last;
    NodePtr curr;
};

typedef struct ListDescr * List;

char s1 [] = "Felix";
char s2 [] = "Heino";
char s3 [] = "Oskar";

struct Node items [] = {
    { items + 1, items + 3, s1 },
    { items + 2, items , s2 },
    { items + 3, items + 1, s1 + 1 },
    { items , items + 2, s3 + 3 }
};

struct ListDescr ld = { items + 1, items , items +2 };

List l = & ld;
```

Welche C-Ausdrücke werden benötigt, um ausgehend von der Variablen *List l* die folgenden Werte zu berechnen. Es sollen möglichst kurze Zugriffspfade gewählt werden.

1. *s1*

   .....

2. das Zeichen *x* aus der Zeichenreihe *Felix*

   .....

3. die Adresse des Zeichens *e* aus der Zeichenreihe *Felix*

   .....

4. das Zeichen *O* aus der Zeichenreihe *Oskar*

   .....

5. die Adresse der Elementes mit dem Wert *k* aus der Zeichenreihe *Oskar*

   .....

6. ein Test, ob *curr* auf das letzte Element der Liste zeigt?

   .....

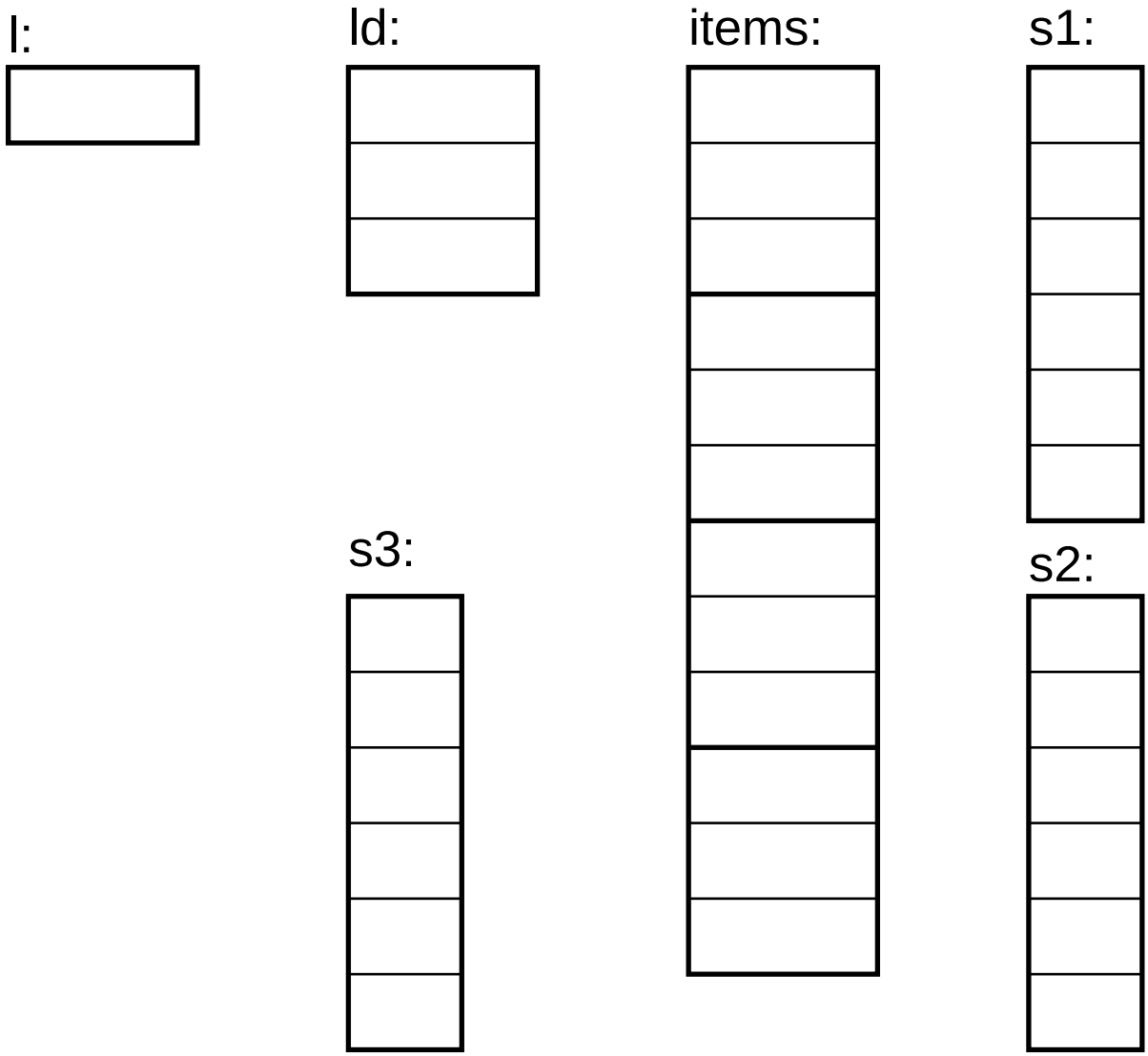
7. ein Test, ob *curr* auf das vorletze Element der Liste zeigt?

   .....

8. die Länge der Zeichenkette des 1. Eintrags in der Liste

   .....

Benutzen Sie in das folgenden Diagramm um zu veranschaulichen, wie die Variablen initialisiert werden.



Aufgabe 5:

Gegeben sei folgendes Pascal Programmstück

```
type
    feld = array [1..10] of integer;

procedure tausch(var f1, f2 : feld);
var
    tmp : feld;
begin
    tmp := f1;
    f1 := f2;
    f2 := tmp
end ;
```

Übersetzen Sie die Typdefinition und die Prozedur in gleichwertige Datentypen und Routinen in C. Verwenden Sie keine Schleifen und kein *memcpy*.

(Ergebnis bitte auf der folgenden Seite)

Das gleichwertige C Programm

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .