Universität Mannheim
Fakultät für Mathematik und Informatik
Lehrstuhl für Praktische Informatik IV
Professor Dr. W. Effelsberg

Teilprüfung
Software- und Internet-Technologie
September 2003: Programmierkurs 2

Name: .... Vorname: ....

Matrikel-Nr.: .... Semester: .... Fach: ....

Hinweise:

1. Bitte füllen Sie sofort den Kopf des Deckblattes aus.
2. Unterschreiben Sie die Klausur auf der letzten Seite.
3. Überprüfen Sie bitte Ihr Klausurexemplar auf Vollständigkeit (**9** Seiten).
4. Tragen Sie die Lösungen – soweit möglich – direkt in die Klausur ein.
5. Zugelassene Hilfsmittel: nicht programmierbarer Taschenrechner
6. Bearbeitungszeit: 66 Minuten.

Aufgabe	max. Punktzahl	Punkte
1	12	
2	15	
3	13	
4	15	
5	11	
Summe	66	

Aufgabe 1: Verständnisfragen *[12 Punkte]*

a) [3 Punkte] Betrachten Sie die folgende rekursive Funktion:

```
int square(int a) {
  if(a = 1)
    return 1;
  else
    return (square(a-1) + 2*a -1;
}
```

Welche Arten von Fehlern (syntaktische, semantische, logische) enthält das Programm? Begründen Sie ihre Antwort!

b) [3 Punkte] Gegeben sei die folgende Variablendeklaration:

```
unsigned int a = 11;
unsigned int b = 16;
unsigned int c;
```

Welche Werte nimmt `c` an für

i) `c = a & b;`

ii) `c = a | b;`

iii) `c = a ^ b;`

c) [3 Punkte] Gegeben sei das folgende Unterprogramm:

```
void doodle(int max, int feld[5]){
  int i;
  for(i = 0; i < max; i++)
    feld[i] *= 2;
}
```

Das Hauptprogramm ruft die Funktion wie folgt auf:

```
int max = 3;
int a[5] = {1,1,1,1,1};
doodle(4,a);
```

Welche Belegung hat das Feld `a[]` nach dem Aufruf der Funktion `doodle()`?

d) [3 Punkte] Das folgende Unterprogramm für M68000-Assembler vertauscht den Inhalt der beiden Register `D0` und `D1`:

```
SWAPBYTE  MOVE.B  D0,D2
          MOVE.B  D1,D0
          MOVE.B  D2,D1
          RTS
```

Ergänzen Sie das Programm so, dass der ursprüngliche Inhalt von Register `D2` für das aufrufende Programm nicht verloren geht. Nutzen Sie dazu den Stack!

Aufgabe 2: C-Programmierung *[15 Punkte]*

Ein Schachbrett besteht aus 8x8 Feldern, die der Einfachheit halber von $(0, 0)$ bis $(7, 7)$ durchnummeriert seien.

Ein Springer ist eine Figur, die sich in einem Zug um 2+1 Felder bewegen darf und zwar

- erst 2 Felder waagerecht, dann 1 Feld senkrecht oder
- erst 2 Felder senkrecht, dann 1 Feld waagerecht.

Ein Springer, der in der Mitte des Schachbrettes steht, hat also 8 mögliche Züge zur Verfügung. Steht er zu nahe am Rand, so scheiden alle Züge aus, für die er das Schachbrett verlassen müsste.

Schreiben Sie ein vollständiges C-Programm, das folgendes leistet:

1. Einlesen der aktuellen Springerposition (x, y) von der Standardeingabe. Bei fehlerhafter Eingabe wird das Programm beendet.
2. Ausgabe aller möglichen Züge des Springers auf der Standardausgabe, ebenfalls in der Form (x, y).

Bsp.:

```
Position des Springers:
(3,0)
Moegliche Zuege:
(1,1)
(2,2)
(4,2)
(5,1)
```

(Platz zum Lösen von Aufgabe 2)

Aufgabe 3: Programmierung mit Zeigern *[13 Punkte]*

Der Datentyp `knoten` einer verketteten Liste sei wie folgt definiert:

```
typedef struct node{
  int value;
  struct node* next;
} knoten;
```

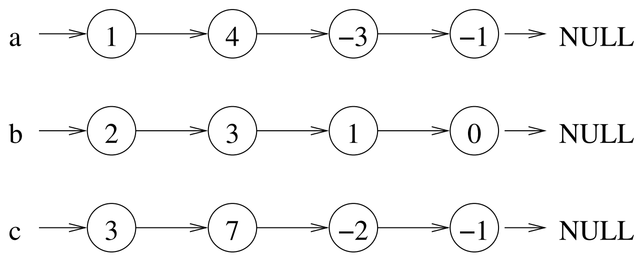
Dieser Datentyp wird nun verwendet, um Integer-Vektoren beliebiger Länge zu implementieren.

Schreiben Sie eine Funktion

```
knoten* summe(knoten* a, knoten* b) {...}
```

die zwei solche Vektoren als Eingabe erhält und als Ergebnis den Summenvektor zurückgibt. Dabei darf ohne Prüfung davon ausgegangen werden, dass die Vektoren a und b die gleiche Länge haben.

Bsp.: `c = summe(a,b);`



(Platz zum Lösen von Aufgabe 3)

Aufgabe 4: Assembler-Programmierung *[15 Punkte]*

Eine Permutation $p : \{0, \ldots, 255\} \to \{0, \ldots, 255\}$ sei definiert durch ihre Wertetabelle (siehe nebenstehendes Beispiel). Eine solche Wertetabelle stehe im Speicher an den Adressen `$7000` bis `$70FF`. Dabei steht an der Stelle `$7000` der Wert $p(0)$, an `$7001` der Wert $p(1)$ usw.

i	Adresse	$p(i)$
0	`$7000`	21
1	`$7001`	20
2	`$7002`	41
⋮	⋮	⋮

Schreiben Sie ein Programm in M68000-Assembler, das aus dieser Tabelle die umgekehrte Permutation p^{-1} berechnet und diese im Speicher unter den Adressen `$7100` bis `$71FF` ablegt (siehe nebenstehendes Beispiel).

i	Adresse	$p^{-1}(i)$
⋮	⋮	⋮
20	`$7114`	1
21	`$7115`	0
⋮	⋮	⋮
41	`$7129`	2
⋮	⋮	⋮

Bem.: Es ist <u>kein</u> vollständiges Assembler-Programm gefragt. Insbesondere müssen keine Assemblerdirektiven (wie `ORG $0` usw.) angegeben werden!

Das Programm ist ausführlich zu kommentieren; unkommentierte Programme werden nicht bewertet!

Aufgabe 5: Rekursive Funktionen *[11 Punkte]*

Gegeben sei die folgende rekursive Funktion in C:

```
int fakultaet(int n) {
  if((n < 1) || (n > 12))
    return -1;
  if(n == 1)
    return 1;
  else
    return (n*fakultaet(n-1));
}
```

Diese Funktion berechnet für $1 \leq n \leq 12$ die Fakultät $n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1$ des übergebenen Parameters, für andere Werte von n gibt sie die Fehlermeldung -1 zurück.

Schreiben Sie eine gleichwertige rekursive Funktion in M68000-Assembler. Der Parameter n soll dabei beim Aufruf im Register `D0` stehen, das Ergebnis soll über Register `D1` zurückgegeben werden.

Das Assembler-Programm ist ausführlich zu kommentieren; unkommentierte Programme werden nicht bewertet!