Lehrstuhl für Praktische Informatik IV
Prof. Dr. Wolfgang Effelsberg
A5, 6, Raum B 223
68131 Mannheim
Telefon: (0621) 181–2600
Email: effelsberg@informatik.uni-mannheim.de

Robert Schiele
B6, 29, Raum C0.04
68131 Mannheim
Telefon: (0621) 181–2214
Email: rschiele@uni-mannheim.de

UNIVERSITÄT MANNHEIM

Programmierkurs II — Klausur
Sommersemester 2006 — 20. Juli 2006

Hinweise

1. Überprüfen Sie bitte Ihr Klausurexemplar auf Vollständigkeit (10 Seiten).
2. Bearbeiten Sie die Aufgaben *ausschließlich* auf dem Aufgabenblatt der jeweiligen Aufgabe.
3. Schreiben Sie auf jedes Blatt, das bewertet werden soll, oben Ihren Namen und Ihre Matrikelnummer.
4. Verwenden Sie nur dokumentenechte Stifte (z. B. keinen Bleistift) und keine roten Stifte.
5. Es sind keine Hilfsmittel zugelassen.
6. Bearbeitungszeit: 66 Minuten.

Korrekturzeile — Bitte *nicht* ausfüllen!

Aufgabe	1	2	3	4	Summe
Max. Punktzahl	16	18	17	15	66
Erreichte Punktzahl					

Name: Matrikelnummer:

Name: Matrikelnummer:

Aufgabe 1 — 16 Punkte

Aufgabe 1 a) — 4 Punkte

Wandeln Sie in folgendem C-Codefragment die `for`-Schleife in eine äquivalente `while`-Schleife um.

```
1  void magic(int*, int*);

   for (int a = 0, b = 10; a <= 10; ++a, --b) {
       magic(&a, &b);
   }
```

Aufgabe 1 b) — 4 Punkte

Der Operator `[]` von C ist nur eine spezielle Schreibweise anderer Ihnen bekannter Operatoren. Geben Sie an, wie der Ausdruck `a[b]` ohne den Operator `[]` geschrieben werden kann.

Name: Matrikelnummer:

Aufgabe 1 c) 4 Punkte

Gegeben sei die Definition einer C-Struktur in folgender Weise:

```
struct gewinn {
    int monat[12];
    int jahr;
};
```

Geben Sie den Funktionsprototypen für eine Funktion `jahresabschluss` an, welche eine derartige Struktur übergeben bekommt, die Gewinnsumme über alle 12 Monate berechnet und in der Strukturvariablen `jahr` zur weiteren Verwendung abspeichert.

Geben Sie auch an, wie die Funktion für die Variable `x` vom oben genannten Typ aufgerufen werden muss.

Eine Implementation der Funktion braucht *nicht* angegeben zu werden.

Aufgabe 1 d) 4 Punkte

Die beiden Assembleranweisungen

```
mov.b 2(r10), 0(r11)
mov.b 3(r10), 1(r11)
```

können leicht durch eine einzige `mov`-Anweisung ersetzt werden, ohne dass sich dadurch das Ergebnis verändert. Nennen Sie *zwei* Gründe, weshalb das für die entsprechenden Anweisungen `add.b` nicht analog gilt.

Name: Matrikelnummer:

Aufgabe 2 — 18 Punkte

Die Determinante ist eine Funktion, die einer quadratischen Matrix eine Zahl zuordnet.

Die Determinante einer Matrix $n \times n$-matrix A ist dabei bestimmbar als

$$\det(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det(A_{ij})$$

Dabei ist i beliebig gewählt, a_{ij} das Element der i-ten Zeile und j-ten Spalte und A_{ij} diejenige $(n-1) \times (n-1)$-Matrix, die aus A entsteht, wenn die i-te Zeile und j-te Spalte gestrichen wird. Die Determinante einer 1×1-Matrix ist gleich ihrem einzigen Element.

Beispiel: Für die Matrix $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ ist $a_{12} = 2$ und $A_{12} = \begin{pmatrix} 4 & 6 \\ 7 & 9 \end{pmatrix}$. Somit gilt also $\det \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = 1 \det \begin{pmatrix} 5 & 6 \\ 8 & 9 \end{pmatrix} - 2 \det \begin{pmatrix} 4 & 6 \\ 7 & 9 \end{pmatrix} + 3 \det \begin{pmatrix} 4 & 5 \\ 7 & 8 \end{pmatrix} =$ $1(5\det(9) - 6\det(8)) - 2(4\det(9) - 6\det(7)) + 3(4\det(8) - 5\det(7)) = 1(5 \cdot 9 - 6 \cdot 8) - 2(4 \cdot 9 - 6 \cdot 7) + 3(4 \cdot 8 - 5 \cdot 7) = 1(45 - 48) - 2(36 - 42) + 3(32 - 35) = 1(-3) - 2(-6) + 3(-3) = -3 + 12 - 9 = 0$.

Implementiert werden soll eine Funktion, welche die Determinante einer quadratischen Matrix (aus `int`-Werten) berechnet und als Rückgabewert zurückgibt. Die Matrix soll dabei in einem eindimensionalen Feld von `int`-Werten gespeichert werden, in dem zuerst die Zeilenanzahl der Matrix und dann alle Elemente zeilenweise gespeichert sind.

Die Matrix aus dem obigen Beispiel würde also definiert werden als:

```
1  int a[] = {
       3,
       1, 2, 3,
       4, 5, 6,
       7, 8, 9,
6  };
```

Schreiben Sie ein *vollständiges* Programm, welches die Funktion(en) und eine `main()`-Funktion enthält, welches die Funktion mit der obigen Beispielmatrix testet und das Ergebnis auf dem Bildschirm ausgibt.

Hinweise:

- Sie dürfen für die Funktion natürlich auch beliebig viele sinnvolle Hilfsfunktionen definieren.
- Achten Sie bei Speicheranforderungen darauf, den Speicher auch wieder freizugeben.
- Verwenden Sie keine globalen Variablen!
- Kommentieren Sie Ihr Programm sinnvoll!

Name: Matrikelnummer:

Name: Matrikelnummer:

Aufgabe 3 — 17 Punkte

Wenn einfach verkettete Listen auf externe Datenträger ausgelagert werden müssen, bietet es sich aus Effizienzgründen an, mehrere Elemente der Liste jeweils zu Blöcken zusammenzufassen.

Wir betrachten im Folgenden verkettete Listen, die definiert sind als

```
typedef struct blocklist {
    int num;
    int val[MAXVAL];
    struct blocklist* next;
} blocklist;
```

Dabei ist `num` die Anzahl der in diesem Block gespeicherten Elemente, `val` das Feld, welches maximal die durch die Konstante `MAXVAL` bestimmte Anzahl von Elementen aufnehmen kann und `next` ein Zeiger auf den Block, der die Liste fortsetzt, oder ein `NULL`-Zeiger für den letzten Block der Liste.

Aufgabe 3 a) — 5 Punkte

Schreiben Sie eine Funktion, die einen Zeiger auf den ersten Block einer solchen Liste und eine Zahl übergeben bekommt und als Rückgabewert zurückgibt, an welcher Position der Liste diese Zahl zum ersten Mal vorkommt oder -1, falls die Zahl nicht vorkommt. Die Zählung der Position soll bei 0 beginnen.

Name: Matrikelnummer:

Aufgabe 3 b) 6 Punkte

Schreiben Sie eine Funktion, die einen Zeiger auf den ersten Block einer solchen Liste, eine Zahl und eine Position übergeben bekommt. Die Funktion soll die übergebene Zahl in der Liste dann an der übergebenen Position einfügen. Alle nachfolgenden Zahlen sollen sich dadurch dann um eine Position nach hinten verschieben. Kann die Zahl nicht an der gewünschten Position eingefügt werden (zum Beispiel weil die Liste nicht so viele Elemente enthält), so soll die Funktion 0 als Rückgabewert zurückgeben, sonst 1.

Aufgabe 3 c) 6 Punkte

Schreiben Sie eine Funktion, die einen Zeiger auf den ersten Block einer solchen Liste und eine Position übergeben bekommt. Die Funktion soll dann das Element an der übergebenen Position aus der Liste löschen. Kann die Zahl nicht an der gewünschten Position gelöscht werden (zum Beispiel weil die Liste nicht so viele Elemente enthält), so soll die Funktion 0 als Rückgabewert zurückgeben, sonst 1. Sollte durch den Löschvorgang ein Block keine Elemente mehr enthalten, so soll er aus der Liste entfernt und sein Speicher wieder freigegeben werden.

Name: Matrikelnummer:

Aufgabe 4 — 15 Punkte

Entwickeln Sie wie unten näher beschrieben ein Programmfragment in MSP430-Assembler, welches für ein Byte-Feld prüft, ob die darin gespeicherten Zahlen in aufsteigend sortierter Reihenfolge vorliegen.

Die Anfangsadresse des Felds wird in Register `R8` und die Länge des Felds in Register `R9` übergeben. Falls die Liste in sortierter Reihenfolge vorliegt, so soll in Register `R10` die Zahl 0 abgelegt werden, sonst die Position des Elements innerhalb der Liste, welches als erstes die Sortierungsbedingung verletzt, wobei für die Position die Zählung beim ersten Element mit 0 beginnt.

Aufgabe 4 a) — 5 Punkte

Formulieren Sie zuerst einen Algorithmus in *Pseudocode*, der die Bedingung für ein Feld `a[]` mit der Länge `n` prüft und ein Ergebnis in der Form produziert, wie sie nach obiger Beschreibung in Register `R10` abgelegt werden soll.

Name: Matrikelnummer:

Aufgabe 4 b) 10 Punkte

Implementieren Sie nun diesen Algorithmus in einem MSP430-Assembler-Programmfragment. Ein Hauptprogramm, welches das Unterprogramm aufruft, braucht *nicht* angegeben zu werden.

Kommentieren Sie Ihren Code sinnvoll!