Lehrstuhl für Praktische Informatik IV
Prof. Dr. Wolfgang Effelsberg
A5, Raum B223
68131 Mannheim
Telefon: (0621) 181–2600
Email: effelsberg@informatik.uni-mannheim.de

Dirk Stegemann
A5, Raum B125
68131 Mannheim
Telefon: (0621) 181–2667
Email: dirk.stegemann@informatik.uni-mannheim.de

UNIVERSITÄT MANNHEIM

Programmierkurs II für Bachelor SIT — Klausur
Herbstsemester 2006 — 25. Januar 2007

Hinweise

1. Überprüfen Sie Ihr Klausurexemplar auf Vollständigkeit (12 einseitig bedruckte Seiten).
2. Unterschreiben Sie die Klausur auf der Rückseite des letzten Blatts.
3. Tragen Sie Ihre Lösungen direkt in die Klausur ein. Benutzen Sie ggf. auch die Rückseiten der Aufgabenblätter.
4. Schreiben Sie auf jedes Blatt, das bewertet werden soll, oben Ihren Namen und Ihre Matrikelnummer.
5. Verwenden Sie nur dokumentenechte Stifte (z. B. keinen Bleistift) und keine roten Stifte.
6. Es sind keine Hilfsmittel zugelassen.
7. Die Bearbeitungszeit beträgt 66 Minuten.

Korrekturzeile — Bitte *nicht* ausfüllen!

Aufgabe	1	2	3	4	5	Summe
Max. Punktzahl	16	12	18	8	12	66
Erreichte Punktzahl						

Name: Matrikelnummer:

Aufgabe 1 — 16 Punkte

Aufgabe 1 a) — 4 Punkte

Betrachten Sie die folgende fehlerhafte C-Funktion, die genau dann 1 zurückgeben soll, wenn im übergebenen `int`-Feld `a` der Länge `n` die Zahl 42 vorkommt.

```
int occurs(int a[], int n){
  int i=n-1;
  while (i >= 0){
   if (a[--i]=42) return 1;
  }
  return 0;
}
```

Welche Ausgabe produzieren die folgenden Anweisungen?

```
int a[] = {36,17,41,39,42};
printf("%i\n", occurs(a,5));
```

Wie muss die Funktion verändert werden, damit sie das Gewünschte leistet?

Name: Matrikelnummer:

Aufgabe 1 b) 6 Punkte

Wandeln Sie die folgende rekursive C-Funktion in eine äquivalente iterative C-Funktion unter Verwendung einer `while`-Schleife um.

```
int fibonacci(int n) {
  if (n == 0) return 0;
  if (n == 1) return 1;
  return (fibonacci(n-1) + fibonacci(n-2));
}
```

Name: Matrikelnummer:

Aufgabe 1 c) 4 Punkte

Vergleichen Sie die rekursive und die iterative Implementation der Fibonacci-Funktion aus Aufgabenteil (b) bezüglich ihrer Laufzeit.

Aufgabe 1 d) 2 Punkte

Geben Sie eine zu folgender C-Funktion äquivalente C-Funktion an, die keinen `[]`-Operator verwendet.

```
int summe(int a[], int n){
  int i, sum=0;
  for (i=0 ; i < n ; i++) sum+=a[i];
  return sum;
}
```

Name: Matrikelnummer:

Aufgabe 2	12 Punkte

Schreiben Sie ein **vollständiges** C-Programm, das aus dem Start- und dem Endzeitpunkt einer Veranstaltung die Veranstaltungsdauer berechnet.

Gehen Sie dabei wie folgt vor:

- Start- und Endzeitpunkt werden dem Programm jeweils in Form von Stunden und Minuten als Kommandozeilenparameter übergeben. Auf nicht sinnvolle Eingaben soll das Programm mit einer Fehlermeldung reagieren und abbrechen. Gehen Sie davon aus, dass der Benutzer nur ganze Zahlen als Parameter eingibt.

  Hinweis: Zur Umwandlung eines Strings `s` in einen `int` können Sie die Funktion `atoi(const char *s)` verwenden.

- Geben Sie das Ergebnis auf der Standardausgabe aus.

Beispiel:

```
./zeitdiff 10 15 11 45
Die Veranstaltung dauert 1 Stunde(n) und 30 Minute(n).
```

Name: Matrikelnummer:

(Platz für die Lösung von Aufgabe 2)

Name: Matrikelnummer:

Aufgabe 3 — 18 Punkte

Schreiben Sie ein Programm zur Multiplikation zweier `int`-Matrizen A und B. Gehen Sie davon aus, dass beide Matrizen bereits eingelesen wurden und im Speicher liegen. Matrix A sei dabei durch das `int`-Feld der Größe `[0..l-1, 0..m-1]` und Matrix B durch das Feld der Größe `[0..m-1, 0..n-1]` bestimmt. Das Ergebnis der Multiplikation ist eine Matrix C der Größe `[0..l-1, 0..n-1]`.

$$\begin{pmatrix} a_{0,0} & \cdots & a_{0,m-1} \\ \vdots & \ddots & \vdots \\ a_{l-1,0} & \cdots & a_{l-1,m-1} \end{pmatrix} \begin{pmatrix} b_{0,0} & \cdots & b_{0,n-1} \\ \\ \vdots & \ddots & \vdots \\ \\ b_{m-1,0} & \cdots & b_{m-1,n-1} \end{pmatrix} = \begin{pmatrix} c_{0,0} & \cdots & c_{0,n-1} \\ \vdots & \ddots & \vdots \\ c_{l-1,0} & \cdots & c_{l-1,n-1} \end{pmatrix}$$

Die Multiplikation der Matrizen A und B kann mittels der Produktsummenformel durchgeführt werden:

$$c_{i,j} = \sum_{k=0}^{m-1} a_{i,k} \cdot b_{k,j} \text{ für } i \in \{0, \ldots, l-1\} \text{ und } j \in \{0, \ldots, n-1\} \ .$$

Aufgabe 3 a) — 4 Punkte

Implementieren Sie zunächst eine Funktion

```
int** create(int l, int n)
```

die die Ergebnismatrix C dynamisch erzeugt und einen Zeiger auf die erzeugte Matrix zurückliefert.

Name: Matrikelnummer:

Aufgabe 3 b) 4 Punkte

Falls eine der Matrizen A und B die Nullmatrix ist, d.h. alle Einträge sind gleich Null, ist das Produkt von A und B ebenfalls die Nullmatrix. Schreiben Sie daher eine Funktion

```
int isNull(int** A, int l, int n)
```

die genau dann 1 zurückgibt, wenn A die Nullmatrix ist, und 0 sonst.

Aufgabe 3 c) 6 Punkte

Implementieren Sie jetzt eine Funktion

```
int** mult(int** A, int** B, int l, int m, int n)
```

die die Matrizen A und B miteinander multipliziert und einen Zeiger auf die Ergebnismatrix C zurückliefert. Die Matrizen A und B werden hierbei mittels call-by-reference übergeben. Benutzen Sie die Funktion `isNull` aus Aufgabenteil (b), um die Berechnung ggf. abzukürzen.

Name: Matrikelnummer:

Aufgabe 3 d) 4 Punkte

Geben Sie abschließend eine Funktion

```
void destroy(int** C, int l, int n)
```

an, die den zuvor dynamisch allokierten Speicher wieder freigibt.

Name: Matrikelnummer:

Aufgabe 4 — 8 Punkte

Aufgabe 4 a) — 3 Punkte

Ist das MSP430-Assembler Codefragment

```
mov 0(r10),r11
add #1,r10
```

äquivalent zur Anweisung `mov @r10+,r11` (mit **kurzer** Begründung)?

Aufgabe 4 b) — 2 Punkte

Das Register `r10` enthalte den Wert `0x8` und das Register `r11` den Wert `0xf`. Welche Werte haben `r10` und `r11` nach Ausführung der folgenden Anweisungen?

```
push r10
push r11
add #1,r11
pop r10
pop r11
```

Aufgabe 4 c) — 3 Punkte

In der Vorlesung wurde erwähnt, dass MSP430-Assemblerbefehle immer an geraden Speicheradressen im RAM abgelegt werden. Gilt auch umgekehrt, dass innerhalb eines MSP430-Assemblerprogramms an jeder geraden Speicheradresse ein Assemblerbefehl beginnt (mit **kurzer** Begründung)?

Name: Matrikelnummer:

Aufgabe 5 — 12 Punkte

Um festzustellen, ob eine gegebene Zahl x in einem aufsteigend sortierten `int`-Feld $A[0 \ldots n-1]$ enthalten ist, kann man folgende C-Funktion benutzen:

```
int binarySearch(int *A, int n, int x){
  int l=0;
  int r=n-1;
  int m;
  while (l<=r){
    m = (l+r)/2;
    if (A[m]==x) return 1;
    if (x < A[m]) r=m-1;
      else l=m+1;
  }
  return 0;
}
```

Implementieren Sie diese Funktion in einem MSP430-Assembler Unterprogramm in folgender Weise:

- Die Anfangsadresse des Felds wird in Register `R8`, die Länge des Felds in Register `R9` und die gesuchte Zahl x in Register `R10` übergeben. Das Ergebnis soll in Register `R11` abgelegt werden.
- Alle Feldeinträge bestehen aus zwei Bytes.
- Für die ganzzahlige Division durch 2 können Sie die Assembleranweisung `RRA <operand>` benutzen.
- Ein Hauptprogramm, das das Unterprogramm aufruft, braucht nicht angegeben zu werden.
- Kommentieren Sie Ihr Programm sinnvoll!

Beispiel:

Parameter

```
R8 : 0x0300
R9 : #5
R10: #6
```

Eingabefeld

Adresse	Inhalt
0x0300	#1
0x0302	#2
0x0304	#4
0x0306	#6
0x0308	#9

Ergebnis

```
R11 : #1
```

Name: Matrikelnummer:

(Platz für die Lösung von Aufgabe 5)