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

Marcel Busse
A5, Raum B221
68131 Mannheim
Telefon: (0621) 181–2616
Email: busse@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
Frühjahrssemester 2007 — 31. August 2007

Hinweise

1. Überprüfen Sie Ihr Klausurexemplar auf Vollständigkeit (9 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	14	16	8	12	66
Erreichte Punktzahl						

Name: Matrikelnummer:

Aufgabe 1 — 16 Punkte

Aufgabe 1 a) — 5 Punkte

Betrachten Sie die folgende C-Funktion.

```
int f(int* a, int n){
  int i,x;
  if (n<1) return 0;
  x = a[0];
  for (i=1 ; i < n ; i++){
   if (a[i]<x) x = a[i];
  }
  return x;
}
```

Welche Ausgabe produzieren die folgenden Anweisungen?

```
int a[] = {5,4,3,2,1};
printf("%i\n",f(a,4));
```

Aufgabe 1 b) — 2 Punkte

Welche Werte haben die Variablen `a` und `c` nach Ausführung des folgenden C-Codes (mit kurzer Begründung)?

```
int a=1;
int* b=&a;
int c=*b;
*b=2;
```

Name: Matrikelnummer:

Aufgabe 1 c) 4 Punkte

Implementieren Sie die Standard-Stringfunktion `char* strcat(char* s1, char* s2)` aus der C-Headerdatei `<string.h>`, die den String `s2` an den String `s1` anhängt und einen Zeiger auf `s1` zurückliefert. Verwenden Sie keine weiteren String-Funktionen wie bspw. `strlen()`. Sie können davon ausgehen, dass `s1` genügend Platz für die Aufnahme von `s2` bereitstellt.

Aufgabe 1 d) 2 Punkte

Geben Sie beispielhaft einen Aufruf der Funktion `strcat(char* s1, char* s2)` an.

Aufgabe 1 e) 3 Punkte

Der Zugriff auf Elemente eines zweidimensionalen Felds kann in C sehr unterschiedlich erfolgen. Betrachten Sie am Beispiel des Felds `c` die folgende Tabelle und ergänzen Sie die fehlenden Tabellenzellen.

Zugriff auf...	1. Möglichkeit	2. Möglichkeit	3. Möglichkeit
1. Zeile, 1. Spalte	`**c`	`*c[0]`	
i. Zeile, 1. Spalte		`*c[i-1]`	`c[i-1][0]`
1. Zeile, j. Spalte	`*(*c+j-1)`		
i. Zeile, j. Spalte			`c[i-1][j-1]`

Name: Matrikelnummer:

Aufgabe 2 — 14 Punkte

Das Spiel 4-Gewinnt basiert auf einem vertikal aufgestellten $n \cdot m$ großen Spielfeld, in das zwei Spieler, 'x' und 'o', abwechselnd ihre Spielsteine von oben hineinwerfen. Wir gehen vereinfachend davon aus, dass derjenige Spieler gewonnen hat, der als erster in einer Spalte vier direkt übereinander liegende Steine hat.

	o				
	x	x		o	
	x	o	o	x	
x	x	o	o	x	o

Abbildung 1: Beispielfeld für $n = 5$ und $m = 6$

Zur Repräsentation des Spielfelds verwenden wir ein `char`-Feld `feld` mit n Reihen und m Spalten, wobei der Inhalt des Felds in Zeile i und Spalte j durch `feld[i][j]` gegeben ist. Die Spielsteine werden durch die Zeichen `'x'` und `'o'` voneinander unterschieden.

Implementieren Sie eine Funktion `int gewinner(char** feld, int n, int m)`, die das übergebene Spielfeld analysiert und folgende Werte zurückliefert:

- 1, falls `'x'` gewonnen hat,
- 2, falls `'o'` gewonnen hat,
- 0, falls das Spielfeld voll ist, aber weder `'x'` noch `'o'` gewonnen hat
- -1 sonst

Name: Matrikelnummer:

Aufgabe 3 — 16 Punkte

Für die Implementierung einer Warteschlange (Queue) mit Hilfe einer verketteten Liste verwenden wir folgende Datenstruktur:

```
typedef struct node{
  int data;          /* Inhalt des Knotens */
  struct node* next; /* Zeiger auf den Nachfolgerknoten */
} queue_t;

queue_t* queue;
```

Aufgabe 3 a) — 3 Punkte

Implementieren Sie eine Funktion `queue_t* init()`, die eine leere Warteschlange erzeugt und einen Zeiger darauf zurückliefert.

Aufgabe 3 b) — 5 Punkte

Implementieren Sie eine Funktion `void insert(queue_t* queue, int wert)`, die einen `int` am Ende der Warteschlange einfügt.

Aufgabe 3 c) 5 Punkte

Implementieren Sie eine Funktion `int extract(queue_t* queue)`, die das erste Element am Anfang der Warteschlange entfernt und dessen `int`-Wert zurückliefert. Falls die übergebene Warteschlange leer ist, soll die Funktion den Wert `-1` zurückgeben.

Aufgabe 3 d) 3 Punkte

Implementieren Sie eine Funktion `void flush(queue_t* queue)`, die alle Einträge aus der Warteschlange entfernt und den gesamten von der Warteschlange benutzten Speicherplatz freigibt.

Name: Matrikelnummer:

Aufgabe 4 — 8 Punkte

Aufgabe 4 a) — 3 Punkte

Beschreiben Sie **kurz**, welche Aufgabe der Watchdog des MSP430-Mikrocontrollers besitzt.

Aufgabe 4 b) — 2 Punkte

Geben Sie ein Codefragment bestehend aus **höchstens zwei** Anweisungen in MSP430-Assembler an, das dieselbe Wirkung hat wie das folgende Codefragment:

```
mov @r10,r11
add #2, r10
mov @r10,r12
```

Aufgabe 4 c) — 3 Punkte

Vervollständigen Sie das folgende Codefragment in MSP430-Assembler (ohne zusätzliche Anweisungen hinzuzufügen), so dass es die Inhalte der Register `r14` und `r15` vertauscht.

```
push ...
mov r14,r15
pop ...
```

Name: Matrikelnummer:

Aufgabe 5 — 12 Punkte

Implementieren Sie ein MSP430-Unterprogramm, dem eine Liste mit Klausurnoten übergeben wird und das die grüne Leuchtdiode des in der Übung verwendeten MSP430-Sensorknotens einschaltet, wenn mindestens die Hälfte der Teilnehmer die Klausur bestanden hat. Andernfalls soll die rote Leuchtdiode eingeschaltet werden.

Die Anfangsadresse des Felds wird in Register `R8` und die Länge des Felds in Register `R9` übergeben. Alle Feldeinträge bestehen aus einem Byte mit Noten zwischen 1 und 5. Wie üblich hat die Klausur bestanden, wer eine Note aus der Menge $\{1, 2, 3, 4\}$ hat.

Beispiel:

Parameter

```
R8 : 0x0300
R9 : #8
```

Eingabefeld

Adresse	Inhalt
0x0300	#1
0x0301	#5
0x0302	#5
0x0303	#4
0x0304	#2
0x0305	#1
0x0306	#5
0x0307	#5

Ergebnis

```
(rot leuchtet)
```

Ihr Unterprogramm soll die folgende Form besitzen:

```
.good:  ...     ; Ihre Assemblerbefehle
        ...
        ret
```

Ein Hauptprogramm, das das Unterprogramm aufruft, braucht nicht angegeben zu werden.

Hinweise:

- Die drei Leuchtdioden des Sensorknotens werden über das Byte an der (absoluten) Adresse `0x0029` im Speicher des Prozessors gesteuert. Das niederwertigste Bit (Bit 0) an dieser Adresse steuert die rote, das Bit 1 die grüne und das Bit 2 die gelbe Leuchtdiode. Eine Leuchtdiode ist genau dann eingeschaltet, wenn das entsprechende Bit auf 0 gesetzt ist.
- Für die ganzzahlige Division durch 2 können Sie die Assembleranweisung `RRA <operand>` verwenden.

Name: Matrikelnummer:

(Platz für die Lösung der Aufgabe 5)