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
Herbstsemester 2007/2008 — 12. Februar 2008

Hinweise

1. Überprüfen Sie Ihr Klausurexemplar auf Vollständigkeit (11 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) — 3 Punkte

Betrachten Sie die folgende C-Funktion:

```
void f(int i){
  if (i<0) return;
  f(i-1);
  printf("%i\n",i);
}
```

Welche Ausgabe produziert die folgende Anweisung?

```
f(3);
```

Aufgabe 1 b) — 2 Punkte

Welche Werte haben die Variablen `a` und `b`, nachdem die folgenden C-Anweisungen ausgeführt wurden?

```
int a=5;
int b=7;
int* c=&a;
*c=b;
 b=(a==b);
```

Name: Matrikelnummer:

Aufgabe 1 c) 5 Punkte

Implementieren Sie eine C-Funktion

```
void min_max(int* a, int n, int* min, int* max),
```

die das Minimum und das Maximum aus dem übergebenen Array `a` der Größe n zurückliefert.

Name: Matrikelnummer:

Aufgabe 1 d) 3 Punkte

Geben Sie beispielhaft einen Aufruf der Funktion `min_max` aus Aufgabe (c) an und deklarieren Sie alle hierfür notwendigen Variablen.

Aufgabe 1 e) 3 Punkte

Implementieren Sie die C-Funktion

```
int strlen(char* s)
```

aus der Standardbibliothek `<string.h>`, die die Länge des übergebenen Strings `s` (ohne Endezeichen) zurückliefert.

Name: Matrikelnummer:

Aufgabe 2 — 12 Punkte

Beim Sudoku-Spiel verwendet man ein quadratisches Spielbrett der Größe 9×9, das in jeweils 9 Quadrate der Größe 3×3 unterteilt ist. Ziel des Spiels ist es, in jedes Kästchen eine Zahl zwischen eins und neun zu schreiben, so dass in keiner Zeile und keiner Spalte und keinem 3×3-Block eine Zahl doppelt vorkommt.

Die folgende Abbildung zeigt solch ein Spielfeld vor Spielbeginn.

			8	5		4		7
4			2			8		
	1			4	3	6		
								6
					6	3		4
						5		
7	8		5					
1								
3	6			8	2	1	5	

Wir gehen davon aus, dass das Spielfeld mittels folgender Anweisungen erzeugt wurde:

```
int i, j;
int** a = (int**) malloc(9 * sizeof(int*));
for (i = 0; i < 9; i++) {
  a[i] = (int*) malloc(9 * sizeof(int));
  for (j = 0; j < 9; j++) a[i][j] = 0;
}
```

Aufgabe 2 a) — 4 Punkte

Implementieren Sie eine C-Funktion

```
void draw(int** a),
```

die das übergebene Spielfeld auf dem Bildschirm ausgibt (ohne Linien). Kästchen `a[i][j]` mit `a[i][j]==0` sollen als leere Kästchen dargestellt werden.

Name: Matrikelnummer:

Aufgabe 2 b) 8 Punkte

Implementieren Sie zur Überprüfung aller Zeilen des Spielfelds eine C-Funktion

```
int error(int** a),
```

die die Nummer der ersten Zeile zurückliefert, die die Sudoku-Bedingung verletzt, d.h. in der mindestens eine Zahl doppelt vorkommt. Falls alle Zeilen in Ordnung sind, soll die Funktion -1 zurückliefern.

Name: Matrikelnummer:

Aufgabe 3 — 18 Punkte

Gegeben sei eine einfach verkettete Liste mit folgender Datenstruktur:

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

list_t* list;
```

Aufgabe 3 a) — 4 Punkte

Implementieren Sie eine C-Funktion

```
list_t* insert(list_t* list, int wert),
```

die den Wert `wert` am Ende der Liste `list` einfügt und einen Zeiger auf das erste Listenelement zurückliefert. Falls `list==NULL` übergeben wird, soll die Funktion zunächst eine neue Liste anlegen.

Name: Matrikelnummer:

Aufgabe 3 b) 7 Punkte

Implementieren Sie eine C-Funktion

```
list_t* delete_even(list_t* list),
```

die aus der übergebenen Liste `list` alle Knoten mit einem geraden `data`-Eintrag herauslöscht und einen Zeiger auf das erste Listenelement zurückliefert.

Aufgabe 3 c) 7 Punkte

Implementieren Sie eine C-Funktion

```
list_t* reverse(list_t* list),
```

die die übergebene Liste `list` umkehrt und einen Zeiger auf das erste Element der umgekehrten Liste zurückliefert.

Name: Matrikelnummer:

Aufgabe 4 — 8 Punkte

Aufgabe 4 a) — 2 Punkte

Fügen Sie in das folgende MSP430-Assembler-Codefragment **genau eine** Anweisung ein, so dass es die Werte von `R10` und `R11` vertauscht.

```
push R10
...
pop R11
```

Aufgabe 4 b) — 2 Punkte

Welche Werte haben die Register `R10` und `R11` nach Ausführung der folgenden MSP430-Assembleranweisungen?

```
mov #17, 0(R12)
mov  #7, 2(R12)
mov @R12+, R11
mov @R12, R10
```

Aufgabe 4 c) — 2 Punkte

Welche Zahl steht nach Ausführung der folgenden MSP430-Assembleranweisungen in Register `R11`?

```
mov.b #1, 0(R10)
mov.b #1, 1(R10)
mov @R10, R11
```

Aufgabe 4 d) — 2 Punkte

Erläutern Sie **kurz**, welche Aufgabe das Befehlszählerregister (Program Counter) des MSP430-Mikroprozessors besitzt.

Name: Matrikelnummer:

Aufgabe 5 — 12 Punkte

Um für natürliche Zahlen a, b, n den Wert von $a^b \bmod n$ zu bestimmen, kann man folgende C-Funktion verwenden, die den Square-and-Multiply Algorithmus implementiert:

```
int squareAndMultiply(int a, int b, int n){
  int z=1;
  while (b>0){
    if (b%2 == 1)
      z = (z*a) % n;
    a = (a*a) % n;
    b = b / 2;
  }
  return z;
}
```

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

- Der Wert der Variablen a wird der Funktion in Register `R8` übergeben, der Wert von b in `R9` und der Wert von n in `R10`. Das Ergebnis soll in `R11` abgelegt werden.
- Nehmen Sie an, dass eine Funktion `mul` zur Verfügung steht, die für zwei natürliche Zahlen x und y den Wert von $x \cdot y$ berechnet. Diese Funktion erwartet x in Register `R12`, y in Register `R13` und gibt das Ergebnis in Register `R14` zurück.
- Gehen Sie weiterhin davon aus, dass eine Funktion `mod` zur Verfügung steht, die für zwei natürliche Zahlen x und y den Wert von $x \bmod y$ berechnet. Diese Funktion erwartet x in Register `R12`, y in Register `R13` und gibt das Ergebnis in Register `R14` zurück.
- Für die ganzzahlige Division durch 2 können Sie die Assembleranweisung `RRA <operand>` verwenden.

Name: Matrikelnummer:

(Platz für die Lösung von Aufgabe 5)