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

UNIVERSITÄT MANNHEIM

Programmierkurs in C für Bachelor IMI — Klausur
Herbst-/Wintersemester 2007 — 12. Februar 2008

Hinweise

1. Überprüfen Sie bitte Ihr Klausurexemplar auf Vollständigkeit (6 Seiten).
2. Unterschreiben Sie die Klausur auf der Rückseite des letzten Blatts.
3. Bearbeiten Sie die Aufgaben *ausschließlich* auf dem Aufgabenblatt der jeweiligen Aufgabe. Benutzen Sie ggf. auch die Rückseite 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 33 Minuten.

Korrekturzeile — Bitte *nicht* ausfüllen!

Aufgabe	1	2	3	Summe
Max. Punktzahl	8	12	13	33
Erreichte Punktzahl				

Name: Matrikelnummer:

Aufgabe 1 — 8 Punkte

Aufgabe 1 a) — 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.

Lösung:

```
void min_max(int* a, int n, int* min, int* max)
  int i;
  if (n == 0 || a == NULL || min == NULL || max == NULL) return;
  *min = *max = a[0];
  for (i = 1; i < n; i++) {
    if (a[i] < *min) *min = a[i];
    if (a[i] > *max) *max = a[i];
  }
}
```

Aufgabe 1 b) — 3 Punkte

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

Lösung:

```
int min, max;
int [] a = {5,4,2,3,1};
min_max(a, 5, &min, &max);
```

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.

Name: Matrikelnummer:

			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.

Lösung:

```
void draw(int** a) {
  int i, j;
  for (i = 0; i < 9; i++) {
    for (j = 0; j < 9; j++) {
      if (a[i][j] != 0) {
        printf("%d ", a[i][j]);
      } else {
        printf(" ");
      }
    }
    printf("\n");
  }
}
```

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.

Lösung:

```
int error(int** a) {
  int i, j;
  int tmp[9];
  for (i = 0; i < 9; i++) {
    for (j = 0; j < 9; j++) tmp[j] = 0;
    for (j = 0; j < 9; j++) {
      if (a[i][j] != 0) {
        if (tmp[a[i][j]-1] != 0) {
          /* error in row i */
          return i;
        }
        tmp[a[i][j]-1] = 1;
      }
    }
  }
  return -1;
}
```

Aufgabe 3 — 13 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) — 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.

Name: Matrikelnummer:

Lösung:

```
list_t* delete_even(list_t* list) {
  list_t* tmp;
  list_t* prev;
  list_t* l = list;
  if (list == NULL) return NULL;
  prev = NULL;
  while (l != NULL) {
    if (l->data % 2 == 0) {
      if (prev != NULL) prev->next = l->next;
      tmp = l->next;
      free(l);
      l = tmp;
    } else {
      prev = l;
      l = l->next;
    }
  }
  return list;
}
```

Aufgabe 3 b) 6 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.

Lösung:

```
list_t* reverse(list_t* list) {
  list_t* tmp;
  list_t* prev = NULL;
  if (list == NULL) return NULL;
  while (list != NULL) {
    tmp = list->next;
    list->next = prev;
    prev = list;
    list = tmp;
  }
}
return prev;
```