Universität Mannheim
Fakultät für Mathematik und Informatik
Lehrstuhl für Praktische Informatik IV
Professor Dr. W. Effelsberg

Teilprüfung
Software- und Internet-Technologie
März 2003: Programmierkurs 2

Name: .... Vorname: ....

Matrikel-Nr.: .... Semester: .... Fach: ....

Hinweise:

1. Bitte füllen Sie sofort den Kopf des Deckblattes aus.
2. Unterschreiben Sie die Klausur auf der letzten Seite.
3. Überprüfen Sie bitte Ihr Klausurexemplar auf Vollständigkeit (**10** Seiten).
4. Tragen Sie die Lösungen – soweit möglich – direkt in die Klausur ein.
5. Zugelassene Hilfsmittel: nicht programmierbarer Taschenrechner
6. Bearbeitungszeit: 66 Minuten.

Aufgabe	max. Punktzahl	Punkte
1	9	
2	10	
3	6	
4	13	
5	12	
6	16	
Summe	66	

Aufgabe 1: Multiple-Choice-Aufgaben *[9 Punkte]*

Beantworten Sie die folgenden Fragen bzgl. der Sprache C (es ist jeweils genau eine Lösung richtig):

a) [1,5 Punkte] Welchen Wert hat `a` nach Ausführung der folgenden Anweisung:

```
int a = 13 | 21;
```

- ☐ 1
- ☐ 5
- ☐ 24
- ☐ 29
- ☐ keinen, da die Operation nicht zulässig ist.

b) [1,5 Punkte] Welchen Dezimalwert hat `a` nach Ausführung der folgenden Anweisung:

```
int a = 0122;
```

- ☐ 82
- ☐ 122
- ☐ 290
- ☐ 1111010
- ☐ keinen, da die Operation nicht zulässig ist.

c) [1,5 Punkte] Welche Ausgabe erzeugt das folgende C-Fragment:

```
int i;
char *s = "klausur";
for(i=1; i<4; i++)
  printf("%c", s[i]);
```

- ☐ `kla`
- ☐ `klau`
- ☐ `lau`
- ☐ `laus`
- ☐ keinen, da die Operation nicht zulässig ist.

d) [1,5 Punkte] Gegeben sei das folgende Unterprogramm:

```
void doodle(int a) {int b = 0; a = 1;}
```

Welchen Wert haben die globalen Variablen `a` und `b` nach Abarbeiten des folgenden Programmfragments:

```
int a = 3, b = 5;
doodle(b);
```

- ☐ `a = 1, b = 0`
- ☐ `a = 1, b = 5`
- ☐ `a = 3, b = 1`
- ☐ `a = 3, b = 5`
- ☐ keinen, da die Operation nicht zulässig ist.

e) [1,5 Punkte] Gegeben sei das folgende Unterprogramm:

```
void verdopple(int feld[5]) {
  int i;
  for(i=4; i> -1; i--)
    feld[i] *= 2;}
```

Welchen Inhalt hat das Feld `test` nach Aufruf des folgenden Programmfragments? Tritt dabei ein Fehler auf?

```
int test[5] = {2,3,4,5,6};
verdopple(test);
```

- ☐ `(2,3,4,5,6)`, logischer Fehler
- ☐ `(4,6,8,10,12)`, kein Fehler
- ☐ `(4,6,8,10,12)`, semantischer Fehler
- ☐ `(4,6,8,10,6)`, semantischer Fehler
- ☐ Inhalt undefiniert, syntaktischer Fehler

f) [1,5 Punkte] Welchen Wert hat `a` am Ende der Schleife?

```
int a=0;
for(; a<20; a += ((a%3 == 0)? 4 : -1));
```

- ☐ 20
- ☐ 21
- ☐ 22
- ☐ keinen (syntaktischer Fehler)
- ☐ keinen (semantischer Fehler)

Aufgabe 2: Summe berechnen *[10 Punkte]*

Schreiben Sie ein vollständiges C-Programm, das die Summe $S(k) = \sum_{i=1}^{k} i$ für $k = 1, \ldots, 50$ am Bildschirm ausgibt, und zwar in der Form

```
S(1) = 1
S(2) = 3
S(3) = 6
S(4) = 10
S(5) = 15
...
```

a) [5 Punkte] Schreiben Sie das Programm unter Verwendung einer `for`-Anweisung.

b) [5 Punkte] Schreiben Sie das Programm ohne Verwendung einer `for`-Anweisung.

Tipp: Man kann natürlich ausnutzen, dass $S(k) = S(k-1) + k$ gilt.

Aufgabe 3: Parameterübergabe in C *[6 Punkte]*

Schreiben Sie eine Prozedur `sort`, die zwei `int`-Variablen so umordnet, dass die erste Variable größer ist als die zweite, also z.B.:

```
a = 5, b = 2   --(sort)-->   a = 5, b = 2
a = 1, b = 3   --(sort)-->   a = 3, b = 1
```

Geben Sie ein Beispiel dafür an, wie diese Funktion vom Hauptprogramm aufgerufen wird!

Aufgabe 4: Dynamische Datenstrukturen *[13 Punkte]*

Eine doppelt verkettete Liste besteht aus Knoten, die ihren Vorgänger und ihren Nachfolger kennen. Eine solche Liste heißt *zyklisch*, wenn das erste Element zugleich Nachfolger des letzten Elementes ist und umgekehrt (siehe Abb. 1).
In jedem Knoten werde ein `unsigned int`-Wert gespeichert. Der Datentyp `knoten` (siehe Abb. 2) ist daher wie folgt definiert:

```
typedef struct node{
  unsigned int value;         // Speichert den int-Wert
  struct node *last;          // Speichert den Vorgaenger
  struct node *next;          // Speichert den Nachfolger
} knoten;
```

a) [6 Punkte] Schreiben Sie eine Funktion mit der Signatur

   `void durchlaufen(knoten* a),`

   die beginnend mit Knoten a die Liste einmal vollständig durchläuft und die Listeninhalte am Bildschirm ausgibt.

b) [7 Punkte] Schreiben Sie eine Funktion mit der Signatur

   `void einfuegen(knoten* a, knoten* b),`

   die den Knoten a hinter dem Listenknoten b in die Liste einfügt, indem sie alle relevanten Vorgänger und Nachfolger korrekt neu verkettet.

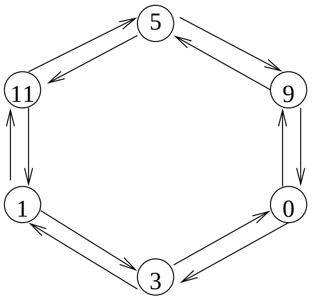

Abb. 1: Aufbau einer zyklischen Liste

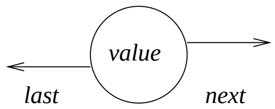

Abb. 2: Datenstruktur Knoten

(Platz zur Lösung von Aufgabe 4)

Aufgabe 5: Assembler-Programmierung (M 68000) *[12 Punkte]*

Schreiben Sie ein möglichst kurzes Programmstück in Assembler, das den Speicherbereich zwischen `$7000` und `$701F` byteweise durchsucht. Nach Abarbeitung des Programmstückes soll

- in Register `D0` der größte Bytewert und
- in Register `D1` der zweitgrößte Bytewert

stehen.

Bem.: Es ist kein vollständiges Assembler-Programm gefragt. Insbesondere müssen keine Assemblerdirektiven (wie `ORG $0` usw.) angegeben werden!

Aufgabe 6: Rekursionen *[16 Punkte]*

Die Quadratfunktion $Q(x) = x^2$ kann (für $x \geq 1$) auch rekursiv definiert werden wie folgt:

$$\begin{aligned} Q(1) &= 1 \\ Q(n) &= Q(n-1) + (2n-1) \end{aligned}$$

a) [6 Punkte] Schreiben Sie eine <u>rekursive</u> Funktion in C mit der Signatur

   `int square(int x),`

   die das Quadrat von x zurückgibt.

b) [10 Punkte] Schreiben Sie ein gleichwertiges, <u>rekursives</u> Programmstück in M68000-Assembler, wobei die Variable x zu Programmbeginn in Register `D0` steht und das Ergebnis $Q(x)$ zu Programmende in Register `D1` stehen soll.

(Platz zur Lösung von Aufgabe 6)