Universität Mannheim
Fakultät für Mathematik und Informatik
Lehrstuhl für Praktische Informatik IV
Prof. Dr. Wolfgang Effelsberg

Teilprüfung
Software- und Internettechnologie
Programmierkurs 2
Wintersemester 2004/2005

Name: .....

Vorname: .....

Matrikel-Nr.: .....

Studienfach: .....

Hinweise:

1. Überprüfen Sie die Klausur auf Vollständigkeit (**11** einseitig bedruckte Seiten).
2. Tragen Sie die Lösungen direkt in die Klausur ein. Benutzen Sie ggf. auch die Rückseiten der Aufgabenblätter.
3. Unterschreiben Sie die Klausur auf dem letzten Blatt.
4. Lösungen auf farbigem Konzeptpapier werden **nicht** bewertet.
5. Zugelassene Hilfsmittel: nicht programmierbarer Taschenrechner
6. Die Bearbeitungszeit beträgt 66 Minuten.

Aufgabe	max. Punktzahl	erreichte Punktzahl
1	13	
2	14	
3	14	
4	10	
5	15	
Summe	66	

Aufgabe 1: Verständnisfragen [13 Punkte]

a) [3 Punkte] Wandeln Sie in folgendem C-Codefragment die `for`-Schleife in eine äquivalente `do-while`-Schleife um:

```
int zahl;

for(;;){
  scanf("Zahl: %i",&zahl);
  if (zahl == 42) break;
}
```

b) [4 Punkte] Welchen Wert haben die Variablen `a`, `b`, `c` und `d` nach Ausführung der folgenden C-Anweisungen?

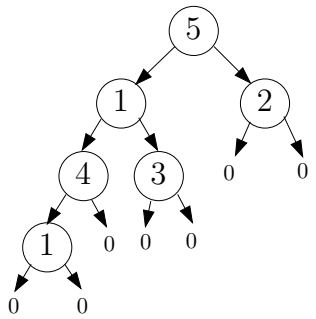
```
unsigned int a = 19;
unsigned int b = 021;
unsigned int c = 0x1f;
unsigned int d;

a = (a^b);
b = (a || c) << 3;
c = a & b;
d = b + c;
```

c) [4 Punkte] Geben Sie einen möglichen Kopf einer C-Funktion `minmax` an, der ein Zeiger auf ein `int`-Feld und die Länge des Felds übergeben werden und die den Wert des kleinsten und den Wert des größten Feldeintrags an das aufrufende Programm zurückliefert. Geben Sie zusätzlich einen beispielhaften Aufruf der Funktion an. Die Angabe des Funktionsrumpfs ist nicht erforderlich.

d) [2 Punkte] Erläutern Sie kurz, welche Aufgabe der Watchdog des MSP430 Mikrocontrollers besitzt.

Aufgabe 2: Dynamische Datenstrukturen [14 Punkte]

In einem Binärbaum haben alle Knoten höchstens zwei Nachfolger, und jeder Knoten außer der Wurzel hat genau einen Vorgänger. In jedem Knoten wird ein `unsigned int` gespeichert.
Der Datentyp `knoten` sei daher wie folgt definiert:




```
typedef struct node{
  unsigned int wert;
  struct node *linkerSohn;
  struct node *rechterSohn;
} knoten;
```

Falls ein Knoten keinen linken bzw. rechten Sohn hat, gilt `linkerSohn==NULL` bzw. `rechterSohn==NULL`.

a) [5 Punkte] Schreiben Sie eine rekursive C-Funktion

```
knoten* vater(knoten* v, knoten* sohn)
```

die im Binärbaum `v` den Vaterknoten des Knotens `sohn` bestimmt. Falls kein Vaterknoten für `sohn` existiert oder eines der beiden Argumente den Wert `NULL` besitzt, soll die Funktion den Wert `NULL` zurückliefern.

b) [5 Punkte] Implementieren Sie eine C-Funktion `void freeBaum(knoten* v)`, die rekursiv den gesamten durch den Binärbaum mit dem Wurzelknoten `v` belegten Speicherplatz freigibt.

c) [4 Punkte] Benutzen Sie die Funktionen aus a) und b), um eine C-Funktion

```
void entferneTeilbaum(knoten* v, knoten* tb)
```

zu implementieren, die den Teilbaum `tb` aus dem Binärbaum mit dem Wurzelknoten `v` entfernt und anschließend den durch `tb` belegten Speicherplatz freigibt.

Aufgabe 3: C-Programmierung [14 Punkte]

Herr Müller will einem Kollegen aus dem linguistischen Institut bei der Analyse von Buchstabenhäufigkeiten in verschiedenen Sprachen helfen. Unterstützen Sie Herrn Müller bei dieser Aufgabe, indem Sie eine C-Funktion

```
void buchstabenstat(FILE* datei)
```

schreiben, die für jeden Buchstaben, d.h. für jedes Zeichen aus $\{a, \ldots, z\}$ (ohne Umlaute) ausgibt, wie oft der Buchstabe in der übergebenen Datei vorkommt und wie groß sein Anteil in Prozent an der Gesamtzahl aller Buchstaben in der Datei ist. Dabei soll nicht zwischen Groß- und Kleinbuchstaben unterschieden werden.

Hinweise:

- Die Funktion `int fgetc(FILE* fp)` liefert das nächste Zeichen aus der Datei, die durch `fp` identifiziert wird, als `unsigned char`. Am Ende der Datei oder bei Lesefehlern ist das Resultat `EOF`.
- Die Funktion `int isalpha(int c)` stellt fest, ob das übergebene Zeichen ein Buchstabe ist.
- Die Funktion `int tolower(int c)` wandelt `c` in einen Kleinbuchstaben um, genauer: Wenn `c` ein Großbuchstabe ist, liefert `tolower(c)` den entsprechenden Kleinbuchstaben, andernfalls ist das Resultat `c`.

(Platz für die Lösung von Aufgabe 3)

Aufgabe 4: Sensorknotensteuerung [10 Punkte]

Schreiben Sie ein **Unterprogramm** in MSP430-Assembler, das in einer Endlosschleife ein Lauflicht mit den Leuchtdioden des in der Übung verwendeten Sensorknotens realisiert. In einem Schleifendurchlauf soll dabei die Lichtfolge rot → gelb → grün → gelb erzeugt werden.

Ihr Unterprogramm soll die folgende Form besitzen:

```
.lights:   ...              ; 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.
- Verwenden Sie die Unterprozedur `wait`, um die Leuchtdauer der Dioden zu steuern. Die Wartedauer wird über die Register `R14` und `R15` übergeben, wobei die höherwertigen 16 Bit der Wartedauer in `R15` erwartet werden.

(Platz für die Lösung von Aufgabe 4)

Aufgabe 5: MSP430-Assembler [15 Punkte]

In dieser Aufgabe soll ein **Unterprogramm** in MSP430-Assembler implementiert werden, das zwei sortierte Integer-Felder zu einem sortierten Integer-Feld zusammenfügt.

Die Anfangsadresse bzw. die Anzahl der Einträge des ersten Felds werden in Register `R8` bzw. `R9` übergeben, die Adresse bzw. die Länge des zweiten Felds in `R10` bzw. `R11`. Die Adresse des Speicherbereichs, in dem das Ergebnis abgelegt werden soll, wird in Register `R12` übergeben. Alle Feldeinträge bestehen aus zwei Bytes.

Beispiel:

Parameter

```
R8 : 0x0300
R9 : #3
R10: 0x0400
R11: #4
R12: 0x500
```

Eingabefelder

Adresse	Inhalt
0x0300	#1
0x0302	#5
0x0304	#8
0x0400	#2
0x0402	#3
0x0404	#5
0x0406	#6

Ergebnisfeld

Adresse	Inhalt
0x0500	#1
0x0502	#2
0x0504	#3
0x0506	#5
0x0508	#5
0x050a	#6
0x050c	#8

Das Unterprogramm soll die folgende Form besitzen:

```
.merge:    ...              ; Ihre Assemblerbefehle
           ...
           ...
           ret
```

a) [5 Punkte] Formulieren Sie zunächst einen Algorithmus in **Pseudocode**, der die Einträge eines sortierten Felds `a[]` der Länge `length(a)` und eines sortierten Felds `b[]` der Länge `length(b)` in einem sortierten Feld `c[]` zusammenfügt.

b) [10 Punkte] Implementieren Sie nun Ihren Algorithmus in einem MSP430-Assembler Unterprogramm. Ein Hauptprogramm, das das Unterprogramm aufruft, braucht nicht angegeben zu werden.

Kommentieren Sie Ihr Programm sinnvoll!