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

Teilprüfung
Software- und Internettechnologie
Programmierkurs 2
Wintersemester 2005/2006

Name: .....

Vorname: .....

Matrikel-Nr.: .....

Studienfach: .....

Hinweise:

a) Überprüfen Sie die Klausur auf Vollständigkeit (**10** einseitig bedruckte Seiten).

b) Tragen Sie die Lösungen direkt in die Klausur ein. Benutzen Sie ggf. auch die Rückseiten der Aufgabenblätter.

c) Unterschreiben Sie die Klausur auf dem letzten Blatt.

d) Zugelassene Hilfsmittel: nicht programmierbarer Taschenrechner

e) Die Bearbeitungszeit beträgt 66 Minuten.

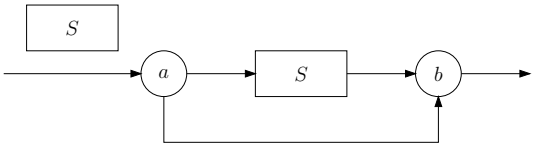
Aufgabe	max. Punktzahl	erreichte Punktzahl
1	13	
2	15	
3	14	
4	10	
5	14	
Summe	66	

Aufgabe 1: Verständnisfragen [13 Punkte]

a) (3 Punkte) Modifizieren Sie das folgende C-Codefragment so, dass es keine `continue`-Anweisung mehr verwendet und das Gleiche leistet.

```
int i;
for (i=0 ; i < 100 ; i++){
  if (i%10 == 0)
    continue;
  printf("%i\n", i);
}
```

b) (2 Punkte) Betrachten Sie das folgende Syntaxdiagramm.



Welche der folgenden Ausdrücke sind korrekt bezüglich S?

(i) `abc`

(ii) `aabb`

(iii) `abba`

(iv) `aab`

c) (5 Punkte) Schreiben Sie eine C-Funktion `copyArray`, mit der man die ersten n Einträge eines `int`-Felds `a` in ein `int`-Feld `b` kopieren kann.

Geben Sie einen beispielhaften Funktionsaufruf für die folgenden Felder an:

```
int a[] = {1,2,3,4,5};
int b[10];
```

d) (3 Punkte) Geben Sie **eine** zu folgendem Codefragment in MSP430-Assembler äquivalente Assembleranweisung an.

```
mov.b 2(r10), 0(r11)
mov.b 3(r10), 1(r11)
```

Aufgabe 2: C-Programmierung [15 Punkte]

Schreiben Sie ein **vollständiges** C-Programm, das für eine reelle Zahl t den Wert $\cos(t)$ mit Hilfe der Formel

$$cos(t) = \sum_{m=0}^{n-1} \frac{(-1)^m}{(2m)!} t^{2m}$$

näherungsweise berechnet. Gehen Sie dabei wie folgt vor:

- Das Argument t und die Anzahl der Iterationen n werden dem Programm als Kommandozeilenparameter übergeben. Bei unzulässigen Eingaben soll das Programm eine Fehlermeldung ausgeben und abbrechen. Gehen Sie davon aus, dass der Benutzer für t nur reelle Zahlen und für n nur ganze Zahlen eingibt.

  Hinweis: Zur Umwandlung eines Strings `s` in einen `int` bzw. `double` können Sie die Funktionen `int atoi(const char *s)` bzw. `double atof(const char *s)` verwenden.

- Berechnen Sie eine Näherung für $\cos(t)$ mit Hilfe der angegebenen Formel. Durch *overflow* bzw. *underflow* enstehende Fehler können hierbei ignoriert werden.

  Zur Erinnerung: Für $m \geq 0$ kann man die Fakultät von m berechnen als

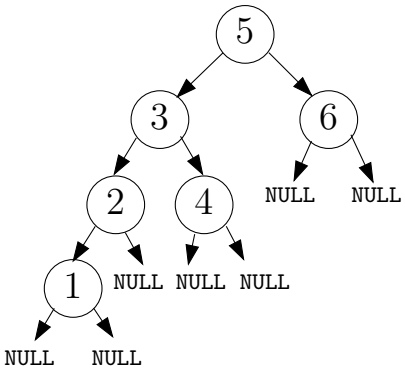
$$m! = \begin{cases} 1 & \text{für} \quad m = 0 \\ (m-1)! \cdot m & \text{für} \quad m > 0 \end{cases}$$

- Geben Sie das Ergebnis der Berechnung über die Standardausgabe aus.

(Platz für die Lösung von Aufgabe 2)

Aufgabe 3: Dynamische Datenstrukturen [14 Punkte]

In einem (geordneten) Binärbaum hat jeder Knoten außer der Wurzel genau eine eingehende Kante und höchstens zwei ausgehende Kanten, die zu seinem linken Sohn bzw. seinem rechten Sohn führen. Jeder Knoten des Binärbaums erfüllt die Ordnungsbedingung, dass sein Inhalt größer oder gleich dem Inhalt seines linken Sohns und kleiner als der Inhalt seines rechten Sohns ist, falls diese Knoten existieren.
In jedem Knoten wird ein `unsigned int` gespeichert. Der Datentyp `knoten` sei daher wie folgt definiert:




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

Falls ein Knoten keinen linken bzw. rechten Sohn hat, gilt `linkerSohn==NULL` bzw. `rechterSohn==NULL`. Ein Knoten heißt Blattknoten, falls er weder einen linken noch einen rechten Sohn hat.

a) [4 Punkte] Schreiben Sie eine C-Funktion

```
knoten* neuerKnoten(unsigned int neuerWert,
                    knoten* links,
                    knoten* rechts)
```

die einen neuen Knoten mit den übergebenen Parametern anlegt und einen Zeiger auf den Knoten zurückliefert. Dabei bezeichnen `neuerWert` den Inhalt des neuen Knotens sowie `links` und `rechts` seinen linken bzw. rechten Sohn. Falls nicht genügend Speicherplatz für den neuen Knoten zur Verfügung steht oder die Parameter die Ordnungsbedingung verletzen, soll die Funktion `NULL` zurückgeben.

b) [4 Punkte] Schreiben Sie eine rekursive C-Funktion

```
unsigned int summe(knoten* b)
```

zur Berechnung der Summe aller im Binärbaum mit der Wurzel `b` gespeicherten Zahlen.

c) [6 Punkte] Implementieren Sie eine rekursive C-Funktion

```
unsigned int tiefe(knoten* b)
```

die die Tiefe des Binärbaums mit der Wurzel `b` berechnet, d.h. die Länge des längsten Pfads von der Wurzel zu einem Blattknoten in `b`. Dabei habe ein Blattknoten die Tiefe 0.

Aufgabe 4: Sensorknotensteuerung [10 Punkte]

Schreiben Sie ein **Unterprogramm** in MSP430-Assembler, das in einer Endlosschleife ein Lauflicht mit der Leuchtfolge rot → gelb → grün mit Hilfe des in der Übung verwendeten Sensorknotens implementiert.

Ihr Unterprogramm soll die folgende Form besitzen:

```
.scan:    ...             ; 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.
- Um die Leuchtzeiten zu steuern, kann die Unterprozedur `wait` verwendet werden. Die Wartedauer wird über die Register `R14` und `R15` übergeben, wobei die höherwertigen 16 Bit der Wartedauer in `R15` erwartet werden.

Aufgabe 5: MSP430-Assembler [14 Punkte]

In dieser Aufgabe soll ein **Unterprogramm** in MSP430-Assembler implementiert werden, das in einem aus Nullen und Einsen bestehenden Byte-Feld die Länge des längsten zusammenhängenden Blocks gleicher Zahlen bestimmt.

Die Anfangsadresse des Felds wird in Register `R8` und die Länge des Felds in Register `R9` übergeben. Das Ergebnis soll in `R10` abgelegt werden. Alle Feldeinträge bestehen aus einem Byte.

Beispiel:

Parameter

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

Eingabefeld

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

Ergebnis

```
R10 : 3
```

a) [4 Punkte] Formulieren Sie zunächst einen Algorithmus in **Pseudocode**, der in einem Feld `a[]` der Länge n den längsten zusammenhängenden Block gleicher Zahlen bestimmt.

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

Kommentieren Sie Ihr Programm sinnvoll!