

**Lösungshinweise zur
Teilprüfung
Software- und Internettechnologie
Programmierkurs 2
Wintersemester 2005/2006**

Aufgabe 1: Verständnisfragen

a)

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

b) Nur (ii) ist korrekt.

c)

```
void copyArray(int *a, int *b, int n){
    int i;
    for (i=0 ; i < n ; i++){
        b[i] = a[i];
    }
}
```

Beispielhafter Funktionsaufruf: `copyArray(a,b,5);`

d) `mov 1(r10), 0(r11)`

Aufgabe 2: C-Programmierung

Aus Zeit- und Genauigkeitsgründen berechnen wir den m -ten Summanden $s(m)$ aus dem Summanden $s(m-1)$. Es gilt

$$\begin{aligned}(-1)^m &= (-1)^{m-1} \cdot (-1) \\ (2m)! &= (2(m-1))! \cdot (2m) \cdot (2m-1) \\ t^{2m} &= t^{2(m-1)} \cdot t^2\end{aligned}$$

und damit $s(m) = s(m-1) \cdot \frac{-1}{(2m) \cdot (2m-1)} t^2$ mit $s(0) = 1$.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv){
    double t, cos_t, s;
    int n, m;

    if (argc != 3){
        printf("Benutzung: cos t n\n");
        exit(1);
    }
    t = (double)atof(argv[1]);
    n = (int)atoi(argv[2]);
    if (n < 1){
        printf("n muss groesser als 0 sein.\n");
        exit(1);
    }

    cos_t=1;
    s = 1;
```

```

for (m=1 ; m < n ; m++){
    s = s * (-1) * t*t / (2*m * (2*m-1));
    cos_t += s;
}

printf("cos(%f) = %f\n",t,cos_t);
return 0;
}

```

Aufgabe 3: Dynamische Datenstrukturen

```

a) knoten* neuerKnoten(unsigned int neuerWert, knoten* links, knoten* rechts){
    knoten *new;
    if ((links!=NULL && links->wert > neuerWert) ||
        (rechts!=NULL && rechts->wert <= neuerWert)) return NULL;
    new = (knoten*) malloc(sizeof(knoten));
    if (new == NULL) return NULL;

    new->wert = neuerWert;
    new->linkerSohn = links;
    new->rechterSohn = rechts;
    return new;
}

b) unsigned int summe(knoten* b){
    if (b==NULL) return 0;
    return summe(b->linkerSohn) + summe(b->rechterSohn) + b->wert;
}

c) unsigned int tiefe(knoten *b){
    unsigned int tiefeLinks, tiefeRechts;
    if (b==NULL || (b->linkerSohn == NULL && b->rechterSohn == NULL)){
        /* b ist leer oder Blatt */
        return 0;
    }

    tiefeLinks = tiefe(b->linkerSohn);
    tiefeRechts = tiefe(b->rechterSohn);
    if (tiefeLinks > tiefeRechts)
        return tiefeLinks + 1;
    else
        return tiefeRechts + 1;
}

```

Aufgabe 4: Sensorknotensteuerung

```

.scan:
    bis.b    #0x7,&0x0029    ; alle LEDs ausschalten
.loop:    bic.b    #0x1,&0x0029    ; rot einschalten
    bis.b    #0x2,&0x0029    ; gruen ausschalten
    mov     #0,r14          ; Wartedauer setzen und warten
    mov     #1,r15
    call    #wait
    bic.b    #0x4,&0x0029    ; gelb einschalten
    bis.b    #0x1,&0x0029    ; rot ausschalten
    mov     #0,r14          ; Wartedauer setzen und warten
    mov     #1,r15
    call    #wait
    bic.b    #0x2,&0x0029    ; gruen einschalten
    bis.b    #0x4,&0x0029    ; gelb ausschalten
    mov     #0,r14          ; Wartedauer setzen und warten
    mov     #1,r15
    call    #wait
    jmp     .loop
    ret

```

Aufgabe 5: MSP430-Assembler

```
a) runSymbol = 2;
   maxRunLen = 0;
   for (i=0 ; i < n ; i++){
       if (a[i]!=runSymbol){
           runSymbol = a[i];
           runLen = 1;
       }
       else
           runLen++;
       if (runLen > maxRunLen)
           maxRunLen = runLen;
   }
   return maxRunLen;
```

```
b) maxRunLen:
    mov     #0,r10           ; maxRunLen = 0
    mov.b  #2,r12           ; runSymbol = a[0]
loop:   tst     r9
        jz     return
        cmp.b  @r8,r12
        jeq   else         ; if a[i] == runSymbol
        mov.b  @r8,r12     ; runSymbol = a[i]
        mov   #1,r11       ; runLength = 1
        jmp   update
else:   add    #1,r11       ; runLen++
update: cmp    r11,r10     ; if runLen > maxRunLen
        jge   inc
        mov   r11,r10     ; maxRunLen = runLen
inc:    add    #1,r8
        sub   #1,r9
        jmp   loop
return: ret
```