Lösungshinweise zur Teilprüfung Software- und Internettechnologie Programmierkurs 2 Frühjahrssemester 2007

Aufgabe 1:

a) Die Funktion sucht nach dem kleinsten Eintrag im übergebenen Feld a der Länge n. Für das gegebene Feld werden nur die ersten vier Elemente untersucht, daher lautet die Ausgabe 2.

b)
```
int a=1;     // a wird 1 zugewiesen
int* b=&a;   // b ist ein Zeiger auf a
int c=*b;    // c wird der Wert von a, d.h. 1, zugewiesen
*b=2;        // a wird 2 zugewiesen
```

Damit gilt am Ende $a = 2$ und $c = 1$.

c)
```
char* strcat(char* s1, char* s2){
  char* start = s1;
  if (s1==NULL || s2==NULL) return start;
  while (*s1!='\0') s1++;
  while (*s2!='\0') *(s1++) = *(s2++);
  *s1 = '\0';
  return start;
}
```

d)
```
char s1[10];
s1[0]='\0';
strcat(s1,"Hello!");
```

e)

Zugriff auf...	1. Möglichkeit	2. Möglichkeit	3. Möglichkeit
1. Zeile, 1. Spalte	`**c`	`*c[0]`	`c[0][0]`
i. Zeile, 1. Spalte	`**(c+i-1)`	`*c[i-1]`	`c[i-1][0]`
1. Zeile, j. Spalte	`*(*c+j-1)`	`*(c[0]+j-1)`	`c[0][j-1]`
i. Zeile, j. Spalte	`*(*(c+i-1)+j-1)`	`*c[i-1]+j-1`	`c[i-1][j-1]`

Aufgabe 2:

```
int gewinner(char** feld, int n, int m){
  int i,j;
  int feld_voll = 1;
  int countX, count0;

  for (j=0 ; j < m ; j++){
    countX = count0 = 0;
    for (i=0 ; i < n ; i++){
      switch (feld[i][j]){
      case 'x':
        count0=0;
        countX++;
        break;
      case 'o':
        countX=0;
        count0++;
        break;
      default:
```

```
        feld_voll = 0;
      } /* switch */
      if (countX==4) return 1;
      if (count0==4) return 2;
    } /* for i */
  } /* for j */

  if (feld_voll){
    return 0; /* Feld voll und keine 4er-Reihe */
  }
  else {
    return -1;
  }
}
```

Aufgabe 3:

a)
```
queue_t* init(){
  queue_t* queue;
  queue = (queue_t*) malloc(sizeof(queue_t));
  if (queue==NULL) return NULL;
  queue->next = NULL;
  return queue;
}
```

b)
```
void insert(queue_t *queue, int wert){
  queue_t* newNode;
  if (queue==NULL) return;

  newNode = (queue_t*) malloc(sizeof(queue_t));
  if (newNode == NULL) return;
  newNode->data = wert;
  newNode->next = NULL;
  while (queue->next != NULL) queue = queue->next;
  queue->next = newNode;
}
```

c)
```
int extract(queue_t* queue){
  queue_t* succ;
  int wert;

  if (queue==NULL || queue->next==NULL) return -1;

  succ = queue->next;
  queue->next = succ->next;
  wert = succ->data;
  free(succ);
  return wert;
}
```

d)
```
void flush(queue_t* queue){
  while (queue!=NULL){
    queue_t* next = queue->next;
    free(queue);
    queue = next;
  }
}
```

Aufgabe 4:

a)
- Abfangen von Programmabstürzen und Endlosschleifen
- Falls die Watchdog-Funktion nicht regelmäßig aufgerufen wird, löst der Watchdog `RESET` aus.

b)
```
mov 0(r10),r11
mov 2(r10),r12
```

c)
```
push r15
mov r14,r15
pop r14
```

Aufgabe 5:

```
.good:
  ;; Parameter:
  ;; r8: Startadresse des Felds
  ;; r9: Laenge des Felds

  ;; lokale Variablen:
  ;; r10: Zaehler fuer die Anzahl der bestandenen Klausuren
  ;; r11: Laufvariable
  ;; r12: Hilfsvariable

  ;; Registerinhalte sichern
       push r8
       push r9
       push r10
       push r11
       push r12

       mov #0, r10          ; Zaehler initialisieren
       mov r9, r11          ; Laufvariable initialisieren

loop:  tst r11         ; r11=0?
       jz end
       cmp.b #5, 0(r8) ; 0(r8) >= #5?
       jge cont
       add #1, r10     ; Zaehler inkrementieren
cont:  add #1, r8      ; Feldzeiger inkrementieren
       sub #1, r11     ; Laufvariable dekrementieren
       jmp loop

end:   bis.b #0x7, &0x0029 ; alle LEDs ausschalten
       mov r10,r12
       add r10,r12
       cmp r9, r12       ; 2*r10=r12 >= r9?
       jge green
       bic.b #1, &0x0029 ; rot einschalten
       jmp final
green: bic.b #2, &0x0029 ; gruen einschalten

final: pop r12           ; Registerinhalte wiederherstellen
       pop r11
       pop r10
       pop r9
       pop r8
       ret
```