Kontrollfragen A

1. Was machen die folgende Zeilen:

```
void xxx(char to[], const char from[])
{
  int i = 0;
  while ((to[i] = from[i]) != '\0')
  {
    i++;
  }
}
```

Kopiert die Zeichenkette (Zeichenvektor) von „from“ nach „to“, solange bis ein \0 kommt.

2. Braucht es zwei Klammerpaare in der while Schleife? Wenn ja, wieso?

Ja, != hat eine höhere Priorität als =.

Kontrollfragen B

1. Definieren Sie einen aufzählenden Typ Color_t, der die drei Werte ROT = 0, BLAU = 5 und GELB = 6 hat.

```
typedef enum Color_
{
 ROT,
 BLAU = 5,
 GELB
} Color_t;
```

2. Wie erzeugen Sie die folgende Ausgabe?

```
Das ist ein Backslash "\".
```

printf("Das ist ein Backslash \"\\\"");

Kontrollfragen C

```
int y, *ip, *iq;     /* ip und iq sind Zeiger auf int Werte          */
int iWert = 22;
ip = &iWert;

*ip = *ip + 10;      /* Erhöhung der int Variabel um 10              */

y = *ip - 1;         /* Wert von *ip minus 1                         */

*ip += 2;            /* int Variabel inkrementieren                  */

++*ip;               /* Vorrang * vor ++ Operator ++(*ip)  (=35)     */
                     /* Erhöhung des Inhaltes wo ip hinzeigt         */

(*ip)++;             /* Klammern notwendig, sonst *(ip++)  (=36)     */

iq = ip;             /* iq zeigt auf Variable wie ip  (*iq ist 36)   */
```

Kontrollfragen D

1. Wieso können Sie die dynamische Speicherallokation auf kleinen, eingebetteten Systemen selten verwenden?

   Weil diese Systeme zu wenig Speicher / Stack haben, sowie auch weil die Funktionen malloc(), free() etc. nicht vorhanden sind.

2. Kreieren Sie dynamisch einen Vektor für 10 double Werte und initialisieren Sie alle Elemente mit dem Wert 1.0.

```
double* dp;
dp = (double*) malloc(sizeof(double) * 10);
if (dp)
{
  int i;
  for(i=0;i<10;i++) {
    *dp(i) = 1.0d;
  }
}

// do something

free(dp);
```