

**Aufgabe 1.** a) Implementiere eine C-Datei zu folgender Header-Datei:

```
1  /* gibt die Laenge eines Strings zurueck */
2  int str_len(char *s);
3
4  /* gibt 0 zurueck, wenn zwei strings gleich
5   * sind und 1 sonst */
6  int str_cmp(char *s1, char *s2);
7
8  /* kopiert s nach d und gibt d zurueck */
9  char *str_cpy(char *d, char *s);
10
11 /* haenge s2 an s1 an und gib s1 zurueck */
12 char *str_cat(char* s1, char* s2)
13
14 /* allokiere neuen Speicher
15  * fuer eine Kopie von s */
16 char *str_dup(char *s);
```

b) Implementiere nun noch folgende (doch eher unübliche) String-Funktionen (ein Palindrom ist ein String, der rückwärts gelesen der Gleiche ist, zum Beispiel "anna"):

```
1  /* schreibt s rueckwaerts in s
2   * und gibt es zurueck */
3  char *str_reverse(char *s);
4
5  /* gibt 1 zurueck, wenn ein String ein Palindrom
6   * ist und 0 sonst */
7  int str_ispalin(char *s);
8
9  /* haenge s2 an s1 an und stelle sicher, dass
10 * dazu genug Speicher in s1 zur
11 * Verfuegung steht */
12 char *str_smartcat(char *s1, char *s2)
13
14 /* verkleinert den Speicher auf den s zeigt
15 * auf die Laenge von s */
16 char *str_compress(char *s);
```

Häufig hilft es auch, sich zu veranschaulichen, wie Funktionen später verwendet werden sollen:

```
1 #include <stdio.h>
2 #include "mystrings.h"
3
4 int main() {
5     char p[100] = "Pepsi_";
6     char c[100] = "Coca_";
7     char suffix[10] = "Cola";
8     char out[100];
9     str_cpy(out,p);
10    str_cat(out,suffix);
11    str_cpy(p,out);
12    str_cpy(out,c);
13    str_cat(out,suffix);
14    str_cpy(c,out);
15    if (str_cmp(p,c)) {
16        printf("%s",p);
17        printf("_is_not_");
18        printf("%s",c);
19        printf("\n");
20    }
21    return 0;
22 }
```

**Aufgabe 2.** Implementiere folgende Funktion (ins gleiche Modul), die zu einem gegebenen String einen längsten Teilstring findet, der ein Palindrom ist. Speichere diesen Teilstring wieder in s und gib s zurück.

```
1 /* Beschreibung, selber machen */
2 char *str_glsp(char *s);
```

Das Problem ist in polynomieller Laufzeit lösbar und auf die Idee kann man auch kommen.

**Aufgabe 3.** Implementiere einige Funktionen um mit quadratischen Matrizen umzugehen:

- a) Eine Funktion, die Speicher für eine quadratische Matrix allokiert, eine um ihn freizugeben, eine um sie auszugeben und eine um sie zur

Einheitsmatrix zu initialisieren (das ist die Matrix mit 1en auf der Hauptdiagonale und 0en sonst):

```
1 double **matrix_alloc(int n);  
2 void      matrix_free(double **A, int n);  
3 void      matrix_print(double **A, int n);  
4 double **matrix_id(double **A, int n);
```

- b) Eine Funktion um eine Matrix zu transponieren (d.h. an der Hauptdiagonale “zu spiegeln”)
- c) Eine Funktion, die zwei solche Matrizen miteinander multipliziert und eine neue Matrix zurück gibt. Für zwei  $n \times n$ -Matrizen  $A = (a_{ij})$  und  $B = (b_{ij})$  ist  $A \cdot B = C = (c_{ij})$  durch  $c_{ij} = \sum_{k=1}^n a_{ik}b_{kj}$  definiert.