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

```c
/* gibt die Laenge eines Strings zurueck */
int str_len(char *s);

/* gibt 0 zurueck, wenn zwei strings gleich
 * sind und 1 sonst */
int str_cmp(char *s1, char *s2);

/* kopiert s nach d und gibt d zurueck */
char *str_cpy(char *d, char *s);

/* haenge s2 and s1 an und gib s1 zurueck */
char *str_cat(char* s1, char* s2)

/* allokiere neuen Speicher
 * fuer eine Kopie von s */
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”):

```c
/* schreibt s rueckwaerts in s
 * und gibt es zurueck */
char *str_reverse(char *s);

/* gibt 1 zurueck, wenn ein String ein Palindrom
 * ist und 0 sonst */
int str_ispalin(char *s);

/* haenge s2 an s1 an und stelle sicher, dass
 * dazu genug Speicher in s1 zur
 * Verfuegung steht */
char *str_smartcat(char *s1, char *s2)

/* verkleinert den Speicher auf den s zeigt
 * auf die Laenge von s */
char *str_compress(char *s);
```

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

```
#include <stdio.h>
#include "mystrings.h"

int main() {
        char p[100] = "Pepsi ";
        char c[100] = "Coca ";
        char suffix[10] = "Cola";
        char out[100];
        str_cpy(out,p);
        str_cat(out,suffix);
        str_cpy(p,out);
        str_cpy(out,c);
        str_cat(out,suffix);
        str_cpy(c,out);
        if (str_cmp(p,c)) {
                printf("%s",p);
                printf(" is not ");
                printf("%s",c);
                printf("\n");
        }
        return 0;
}
```

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.

```
/* Beschreibung, selber machen */
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.