Aufgabe 6.1 Fragen erkennen (Pflicht)

Gegeben sei folgendes Gerüst:

```
int ist_frage(char s[])
{
    // TODO
}

void erkenne_frage(char s[])
{
    printf("(%c) %s\n", " x"[ist_frage(s)], s);
}                    // ^ Leerzeichen!

void aufgabe_6_1()
{
    erkenne_frage("Test");
    erkenne_frage("Hallo!");
    erkenne_frage("Wie geht's?");
    erkenne_frage("Nanu?!");
}
```

Implementiere die Funktion `ist_frage`. Diese Funktion soll den Wert 1 liefern, falls der übergebene String mit einem Fragezeichen endet, ansonsten 0.

Kannst Du Deinem Betreuer das `printf` erklären, insbesondere das vorletzte Argument?

Aufgabe 6.2 Zeichen ersetzen (Pflicht)

Gegeben sei folgendes Gerüst:

```
void ersetze(char s[], char x, char y)
{
    // TODO
}

void zeige_ersetzung(char s[], char x, char y)
{
    printf("========== vorher ==========\n%s\n", s);
    ersetze(s, x, y);
    printf("========= nachher ==========\n%s\n", s);
}
```

```
void aufgabe_6_2()
{
    char a[] = "the goose has nested";
    char b[] = "Veni. Vidi. Vici.";
    zeige_ersetzung(a, ' ', '-');
    zeige_ersetzung(b, '.', '\n');
}
```

Implementiere die Funktion `ersetze`. Diese Funktion soll im String `s` alle Vorkommen des Buchstabens, welcher im Parameter `x` steht, durch denjenigen Buchstaben ersetzen, welcher im Parameter `y` steht.

Aufgabe 6.3 Palindrome erkennen (Pflicht)

Gegeben sei folgendes Gerüst:

```
int ist_palindrom(char s[])
{
    // TODO
}

void erkenne_palindrom(char s[])
{
    printf("(%c) %s\n", " x"[ist_palindrom(s)], s);
}

void aufgabe_6_3()
{
    erkenne_palindrom("anna");
    erkenne_palindrom("aura");
    erkenne_palindrom("rotor");
    erkenne_palindrom("V");
    erkenne_palindrom("rentner");
    erkenne_palindrom("rentier");
    erkenne_palindrom("sabat");
    erkenne_palindrom("lagerregal");
}
```

Implementiere die Funktion `ist_palindrom`. Diese Funktion soll den Wert 1 liefern, falls der übergebene String ein Palindrom ist (d.h. wenn er vorwärts und rückwärts gelesen dasselbe Wort ergibt), ansonsten 0.

Aufgabe 6.4 strlen (Pflicht)

Die Funktion `strlen` bestimmt die Anzahl Zeichen vor der terminierenden NUL:

```
#include <string.h>
…
printf("%d\n", strlen("Hallo"));
5
```

Schreibe eine eigene Funktion `int my_strlen(char a[])`, die dasselbe tut:

```
printf("%d\n", my_strlen("Hallo"));
5
```

Hinweis: Innerhalb von `my_strlen` ist es natürlich verboten, einfach `strlen` aufzurufen, das wäre zu simpel :) Du musst selber nach der terminierenden NUL suchen.

Aufgabe 6.5 strcpy (Kür)

Schreibe eine Funktion `void my_strcpy(char ziel[], char quelle[])`, die alle Zeichen aus Quelle in Ziel hineinkopiert:

```
char a[16];
my_strcpy(a, "Hallo Welt!");
```

Zum Testen wirst Du möglicherweise Strings miteinander vergleichen wollen. Bedenke, dass man Strings nicht mit == vergleichen kann. Damit würde nur verglichen werden, ob die Strings an derselben Speicherstelle liegen. Verwende stattdessen die Funktion `strcmp`.

Aufgabe 6.6 strcat (Kür)

Schreibe eine Funktion `void my_strcat(char ziel[], char quelle[])`, die alle Zeichen aus Quelle an Ziel anhängt:

```
char a[16] = "Hallo ";
my_strcat(a, "Welt!");
```

Aufgabe 6.7 strstr (Kür)

Schreibe eine Funktion `int my_strstr(char a[], char b[])`, die den String `b` im String `a` sucht. Liefere den Index des ersten Vorkommens zurück. Falls `b` nicht in `a` vorkommt, liefere -1 zurück. (Warum kannst Du in diesem Fall nicht 0 zurückliefern?)

Aufgabe 6.8 strcmp (Kür)

Schreibe eine Funktion `int my_strcmp(char a[], char b[])`. Die genaue Spezifikation von `strcmp` findest Du ganz unten auf dem Handout zur Vorlesung.

Teste ausführlich! Was passiert, wenn `a` und `b` verschiedene Längen haben? Was passiert, wenn `a` ein Präfix von `b` ist (zum Beispiel `Welt` und `Weltall`) oder umgekehrt?