Aufgabe 1. Implementiere eine Funktion die zu einem gegebenen Funktionenpointer $f : \mathbb{R} \to \mathbb{R}$, einem Dateinamen, einer Schrittweite $s \in \mathbb{R}$, einer Startstelle x_1 und einer Endstelle x_2 die Wertetabelle der Funktion zwischen x_1 und x_2 zur Schrittweite s speichert. Dabei sollen x und $f(x)$ durch einen Tabulator getrennt werden und jedes Paar $(x, f(x))$ in einer eigenen Zeile stehen. Etwa wäre die Ausgabe für $f = \cos$ zwischen $x_1 = 0$ und $x_2 = 0$ mit Schrittweite $s = 0.1$ die folgende:

```
0.0 1.0
0.1 0.995004165278
0.2 0.980066577841
0.3 0.955336489126
0.4 0.921060994003
0.5 0.87758256189
0.6 0.82533561491
0.7 0.764842187284
0.8 0.696706709347
0.9 0.621609968271
1.0 0.540302305868
```

Aufgabe 2. In dieser Aufgabe geht es um numerische Integration.

a) Implementiere eine Integrationsfunktion, die das Intervall $[a, b]$ in n gleich große Teile aufteilt, für diese jeweils die Trapezsumme (aus der Vorlesung) berechnet und diese aufsummiert:

```
double integrate(double a, double b,
  double (*f)(double), unsigned int n);
```

b) Schreibe nun eine Funktion, die nicht die Anzahl der Teilintervalle erhält, sondern eine "Fehlertoleranz" e. Die Funktion die Aufteilung solange verfeinern, bis sich der approximierte Wert für das Integral durch eine Verfeinerung nur noch um weniger als e ändern würde.

Aufgabe 3. Implementiere folgende Funktion, 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 4. Auf der Homepage gibt es ein Modul `rationals`, in dem rationale Zahlen implementiert sind. Binde es in ein neues Projekt ein und deklariere ein Array von Brüchen, in etwa so:

```
1 RATIONAL arr[7] = {
2  {-1,2}, {1,2}, {3,4}, {9,7}, {10,1}, {7,3}, {11,2} };
```

Nun sortiere `arr` mit Hilfe von `qsort`.

Aufgabe 5. Brainfuck ist eine sogenannte esoterische Programmiersprache, das sind Sprachen, die meist zu wissenschaftlichen oder theoretischen Zwecken, oder einfach zum Spaß entwickelt wurden.

Brainfuck besteht nur aus 8 Befehlen: $> < + - , . [ ]$ alle anderen Zeichen werden als Kommentar interpretiert. Diese Befehle werden, wie bei C auch, nacheinander ausgeführt. Sie operieren auf einem (potentiell unendlich langen) Band (welches aus Zellen besteht in denen jeweils ein `char` steht) indem sie einen Lese-/Schreibkopf über das Band bewegen und Zeichen lesen / schreiben lassen. Das Band ist überall mit `'\0'` vorinitialisiert und der Lese-/Schreibkopf startet an "Position 0" des Bandes. Die Befehle haben folgendee Bedeutung:

$>$ bzw. $<$	schiebt den Lese-/Schreibkopf eins nach rechts bzw. links
$+$ bzw. $-$	in- bzw. dekrementiert den Bandwert unter dem Lese-/Schreibkopf um 1
.	gibt den Wert unter dem Lese-/Schreibkopf aus
,	liest ein Zeichen vom Benutzer ein und schreibt es unter den Lese-/Schreibkopf
[	springt zum zugehörigen]-Befehl, wenn der Wert unter dem Lese-/Schreibkopf 0 ist, sonst soll nichts passieren
]	springt zum zugehörigen [-Befehl, wenn der Wert unter dem Lese-/Schreibkopf verschieden von 0 ist

So sieht ein “Hallo-Welt”-Programm in Brainfuck aus:

```
++++++++++
[
    >+++++++>++++++++++>+++>+<<<<-
]
>++.
>+.
+++++++..
+++.>++.
<<+++++++++++++++.
>.+++.
------.--------.
>+.>.
```

Deine Aufgabe ist es nun, ein Programm zu schreiben, welches Brainfuck-Programme einlesen und ausführen kann.