Aufgabe 1.

a) Implementiere die Addition, Multiplikation, Potenzen und Division komplexer Zahlen. Verwende dazu folgende Header-Datei:

```c
#ifndef _COMPLEX__H
#define _COMPLEX__H

typedef struct _COMPLEX {
        double real;
        double imag;
} COMPLEX;

COMPLEX cplx_add(COMPLEX a, COMPLEX b);
COMPLEX cplx_mul(COMPLEX a, COMPLEX b);
COMPLEX cplx_pow(COMPLEX a, unsigned long n);
COMPLEX cplx_div(COMPLEX a, COMPLEX b);

#endif
```

b) Implementiere die Addition, Multiplikation und Division sowie das Kürzen rationaler Zahlen. Schreibe dazu erst die Header-Datei.

Aufgabe 2. Eine von den beiden Aufgaben vom letzten Zettel fertig machen. Aber dann schon die Listen machen.

FLIP ME

Aufgabe 3. Lese noch einmal im Skript die Sektion 7.5 und implementiere doppelt verkettete Listen, die `double`-Variablen speichern.

```
1  /* Definiere hier angemessene Strukturen für einen
2     einzelnen Listeneintrag und die Liste selbst. */
3
4  /* Leere Liste erstellen */
5  LIST *list_create();
6
7  /* Element hinter E einfügen, NULL heißt am Anfang */
8  LISTNODE *list_insert(LIST *L, LISTNODE *E, double p);
9
10 /* Element am Anfang bzw. Ende einfügen */
11 LISTNODE *list_unshift(LIST *L, double p);
12 LISTNODE *list_push(LIST *L, double p);
13
14 /* Element am Anfang bzw. Ende entfernen und
15    die Daten zurück geben */
16 double list_shift(LIST *L);
17 double list_pop(LIST *L);
18
19 /* eine Element aus der Liste entfernen */
20 void list_delete(LIST *L, LISTNODE *E);
21
22 /* zwei Listen zusammenfügen */
23 LIST *list_merge(LIST *L, LIST *M);
24
25 /* Liste inklusive allen Elementen frei geben */
26 void list_free(LIST *L);
```