Aufgabe 1. a) Implementiere die Addition, Multiplikation, Potenzen und Division komplexer Zahlen.

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

Aufgabe 2. Implementiere „dynamische Arrays“. Also Funktionen, die es leicht ermöglichen mit dynamisch großen Arrays zu arbeiten. Verwende – wenn du möchtest – eine Header-Datei in folgendem Stil:

```c
typedef struct {
  double *data; /* eigentliches Array */
  int length;   /* erstes nicht verwendetes Element */
  int _size;    /* Menge der allokierten Elemente */
} DBLARRAY;

/* initialisiert eine Array-Datenstruktur */
DBLARRAY *dblarray_init();

/* gibt eine Array-Datenstruktur wieder frei */
void dblarray_free(DBLARRAY *);

/* setzt den Wert an der Stelle i auf val
 * falls noetig wird neuer Speicher allokiert und
 * alle Elemente bis dorthin mit 0 initialisiert */
int dblarray_set(DBLARRAY *, int i, double val);

/* setze das erste nicht initialisierte Element auf
 * val sollte es noch nicht existieren wird neuer
 * Speicher allokiert */
int dblarray_push(DBLARRAY *, double val);

/* gib den Wert an der Stelle i zurueck */
double dblarray_get(DBLARRAY *, int i);
```

Es ist gelegentlich günstig, sich vorher zu überlegen, wie man ein Modul verwendet, bevor man es implementiert. Schreibe dir also eine passende `main.c`, die dein Modul benutzt um es zu testen.

Aufgabe 3. Implementiere doppelt verkettete Listen, die `double`-Variablen speichern.

```
/* Definiere hier angemessene Strukturen fuer einen
   einzelnen Listeneintrag und die Liste selbst. */

/* Leere Liste erstellen */
LIST *list_create();

/* Element hinter E einfuegen, NULL heisst am Anfang */
LISTNODE *list_insert(LIST *L, LISTNODE *E, double p);

/* Element am Anfang bzw. Ende einfuegen */
LISTNODE *list_unshift(LIST *L, double p);
LISTNODE *list_push(LIST *L, double p);

/* Element am Anfang bzw. Ende entfernen und
   die Daten zurueck geben */
double list_shift(LIST *L);
double list_pop(LIST *L);

/* eine Element aus der Liste entfernen */
void list_delete(LIST *L, LISTNODE *E);

/* zwei Listen zusammenfuegen */
LIST *list_merge(LIST *L, LIST *M);

/* Liste inklusive allen Elementen frei geben */
void list_free(LIST *L);
```