Aufgabe 1. 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);
```

FLIP ME

Aufgabe 2. Implementiere „dynamische Arrays“ von `double`-Variablen. Also Funktionen, die es leicht ermöglichen mit dynamisch großen Arrays zu arbeiten. Verwende 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. Liefert 1
   bei Erfolg und andernfalls 0. */
int dblarray_init(DBLARRAY *, int initial_size);

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

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

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

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

/* Liefere das letzte initialisierte Element und
   verringere die Länge des Arrays um 1. */
double dblarray_pop(DBLARRAY *);
```

Aufgabe 3. Schreibe Funktionen, die aus einer verketteten Liste ein `DBLARRAY` und umgekehrt machen.