Mache zuerst den Zettel von Freitag fertig.

Aufgabe 1. Berechne von Hand in jeder Zeile die Werte der Variablen und Zeiger, die sich verändern.

```
1 int a = 2, b = 5, *c = &a, *d = &b;
2 a = *c * *d;
3 *d -= 3;
4 b = a * b;
5 c = d;
6 b = 7;
7 a = *c + *d;
```

Am Ende sollte hier herauskommen, dass $a = 14$ ist.

Aufgabe 2. Schreibe ein Modul `arrayhelpers`, das einige nützliche Funktion zum `int`-Array-Handling enthält. Dieses Modul wird sowohl mit statischen, als auch dynamischen Arrays funktionieren.

a) Speicher für ein dynamisches Array der Länge n allokieren.

b) Den Speicher eines dynamischen Arrays freigeben.

c) Array zeilenweise oder mit Kommata getrennt ausgeben.

d) Array sortieren.

e) Alle Felder eines Arrays mit einem Wert initialisieren.

f) Array um 1 rotieren (d.h. das hinterste Element an erste Stelle schreiben und alle anderen Elemente um eins nach hinten schieben).

g) Array um k rotieren.

h) Array umdrehen.

i) Ein Array in einem anderen suchen und die Position zurück geben. Sollte das Array nicht im anderen enthalten sein, so soll der Rückgabewert -1 sein.

Beispiel:

```c
int A[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int B[3] = {4, 5, 6}
int C[2] = {5, 7}
int D[2] = {9, 10}
```

Hier gilt: `B` ist an 3-ter Stelle in `A` enthalten und `D` an 8-ter. Das Array `C` ist garnicht in `A` enthalten, darum wird der Rückgabewert -1 sein.

Aufgabe 3. Füge die Funktionen von Zettel 5 dem Modul `arrayhelpers` hinzu und teste dein Modul ausgiebig. Benutze dabei sowohl statische als auch dynamische Arrays.

Aufgabe 4. Implementiere einige Funktionen um mit Vektoren $v \in \mathbb{R}^n$ umzugehen:

a) Eine Funktion, die Speicher für einen Vektor allokiert, eine um ihn freizugeben, eine um den Vektor auszugeben und eine um ihn zu allokieren und direkt als Nullvektor zu initialisieren (ein Vektor dessen Einträge alle 0 sind).

```c
double *vektor_alloc(int n);
void    vektor_free(double *vektor, int n);
void    vektor_print(double *vektor, int n);
double *vektor_null(double *vektor, int n);
```

b) Eine Funktion um zwei Vektoren zu addieren und das Ergebnis in den zweiten Vektor zu schreiben.

c) Eine Funktion, die das Skalarprodukt zweier Vektoren berechnet.

Aufgabe 5. Diese Aufgabe läuft auf die Implementierung des Merge-Sort Algorithmus hinaus.

a) Implementiere eine Funktion `merge`, die zwei bereits sortierte (eventuell verschieden große) Arrays als Argumente erhält, diese zu einem sortieren Array kombiniert und dieses zurück liefert.

b) Die Funktion `mergesort` selbst soll ein Array in zwei (möglichst gleich große) Teilarrays zerlegen, sich für diese Teilarrays selbst aufrufen und danach die dann sortierten Teilarrays mit der `merge`-Funktion kombinieren.

Erhält die Funktion ein Array mit keinem oder einem Element so belässt es dieses Array wie es ist, dann ist es nämlich bereits sortiert.

Hier als Tipp ein Vorschlag für die Signaturen der beiden Funktionen:

```
1 int *merge(int *list1, int n, int *list2, int m);
2 void mergesort(int *list, int n);
```