**Aufgabe 1.** Schreibe Funktionen `square_to` und `root_to`, die einen `double`-Pointer entgegen nehmen, die dort stehende Variable quadrieren bzw. daraus die Wurzel ziehen und das Ergebnis sowohl zurück geben als auch an die gleiche Speicherstelle schreiben.

**Aufgabe 2.** Schreibe ein Modul `arrayhelpers`, das einige nützliche Funktion zum Umgang mit `int`-Arrays enthält:

a) Array zeilenweise oder mit Kommata getrennt ausgeben

b) Alle Felder eines Arrays mit einem Wert initialisieren

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

d) Array um $k$ rotieren

e) Array umdrehen

f) 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:*

```
1 int A[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
2 int B[3] = {4, 5, 6}
3 int C[2] = {5, 7}
4 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.** In dieser Aufgabe geht es um Sortieralgorithmen. Definiere dir ein Test-Array mit einer *festen* Anzahl von Einträgen mit denen du den Algorithmus testest.

a) Implementiere folgenden Sortieralgorithmus: Sortiere das kleines Element an die erste Stelle, dann das zweitkleinste Element an die zweite Stelle usw.

Oder anders ausgedrückt: Betrachte das Array von Eintrag 1 bis Eintrag $n$. Suche das kleinste Element und vertausche es mit dem ersten

Element des Arrays. Wiederhole dieses Vorgehen dann für das Array von Eintrag 2 bis Eintrag $n$ usw.

b) Der obige Sortieralgorithmus hat Komplexität $\mathcal{O}(n^2)$ (wobei $n$ die Anzahl der Elemente ist). Aus theoretischer Sicht sind Sortieralgorithmen bis zu $\mathcal{O}(n \log(n))$ realisierbar. Wenn man nun aber die größer der zu sortierenden Eintrag einschränkt (z.B. sei die größte zu sortierende Zahl 20000) ist es sogar möglich einen *linearen* Sortieralgorithmus zu implementieren, also $\mathcal{O}(n)$. Dazu stellt man sich für jede Zahl einen leeren "Bucket" (Korb) vor. Dann geht man die Liste der zu sortierenden Einträge durch und für ein Verkommen der Zahl $k$ einen Ball in den $k$-ten Bucket. Danach geht man die Buckets vom ersten bis zu letzten durch. Ist am $k$-ten Bucket angekommen und es liegen $j$ Bälle darin, dann schreibe sukzessive $j$ mal die Zahl $k$ in die zu sortierende Liste. Da die Bälle genau den zu sortierenden Zahlen entsprechen ist die Liste nachher sortiert.

Du kannst natürlich beide Algorithmen in das oben geschriebene Modul auslagern