Aufgabe 1. Implementiere eine c-Datei zu folgender Header-Datei:

```c
/* Gibt die Länge eines Strings zurück. */
int str_len(char *s);

/* Gibt 0 zurück, wenn zwei strings gleich
   sind und 1 sonst. */
int str_cmp(char *s1, char *s2);

/* Kopiert s nach d und gibt d zurück. Es ist nicht
   Aufgabe von str_cpy, dafür zu sorgen, dass bei d
   genug Platz für s ist. */
char *str_cpy(char *d, char *s);

/* Hänge s2 and s1 an und gib s1 zurück. Es ist nicht
   Aufgabe von str_cat, dafür zu sorgen, dass hinter
   s1 genügend legal  beschreibbarer Speicher ist. */
char *str_cat(char* s1, char* s2)

/* Allokiere neuen Speicher für eine Kopie von s. */
char *str_dup(char *s);

/* Falls der string needle im string haystack an
   der stelle i das erste mal vorkommt, gib den
   pointer haystack+i zurück.
   Andernfalls, liefere NULL. */
char *str_str(char *haystack, char *needle);
```

Aufgabe 2. Implementiere den sogenannten *Bucket-Sort* Algorithmus für arrays `arr` von `short int`:

a) Finde das Maximum `m` aus `arr`.

b) Erstelle dann ein `unsigned`-Array `buckets` mit `m+1` Elementen und sorge dafür, dass an der `i`-ten Stelle die Anzahl der `i`'s steht, die in `arr` vorkommen.

c) Iteriere über `buckets` und sortiere zurück nach `arr`: Schreibe dabei `k` mal das Element `i`, wenn an der `i`-ten Stelle von `buckets` ein `k` steht.

Aufgabe 3. Schreibe ein Modul `arrayhelpers`, das einige nützliche Funktion zum `int`-Array-Handling enthält:

a) Array zeilenweise oder mit Kommata getrennt ausgeben

b) Array sortieren

c) Alle Felder eines Arrays mit einem Wert initialisieren

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

e) Array um k rotieren

f) Array umdrehen

g) 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 4. * Diese Aufgabe ist schwierig, lass dir helfen, wenn du sie lösen möchtest :

a) Schreibe eine Funktion, die prüft, ob ein Array eine Permutation ist.

b) Schreibe eine Funktion, die ein Array und ein Permutationsarray entgegennimmt und das Array entsprechend permutiert.