Aufgabe 1. Gegeben sei eine Datei, in der ausschließlich Zahlen stehen. In der ersten Zeile stehe eine natürliche Zahl, die angibt wie viele Zahlen noch folgen.

a) Schreibe ein Programm, dass diese Datei einliest, die Zahlen sortiert und die Datei mit der sortierten Liste überschreibt.

b) Modifiziere dein Programm nun so, dass in der ersten Zeile nicht mehr stehen muss wie viele Zeilen noch folgen.

Aufgabe 2. Schreibe ein Programm, welches eineDatei im folgenden Format ausliest:

```
1 sin(0.4) = 0.389
2 sin(0.45) = 0.43496
3 sin(0.6) = 0.5346
```

Wenn eine Zeile einen Fehler enthält, der größer als 10^{-3} ist, so soll dieser Fehler mit Zeilennummer auf der Konsole ausgegeben werden.

Aufgabe 3. Implementiere den Bucket-Sort Algorithmus: Unter der Annahme, dass die zu sortierenden Daten aus einem endlichen Wertebereich stammen kann man dies theoretisch sehr schnell machen (man sagt: in Linearzeit). Ohne große Einschränkung der Universalität des Algorithmus betrachten wir hier nur ein `unsigned short`-Array A. Bucketsort besteht nun aus zwei Durchläufen:

a) Sei n das Maximum aus A, erstelle dann ein `unsigned`-Array B mit $n+1$ Elementen und sorge dafür, dass an der i-ten Stelle die Anzahl der is steht, die in A vorkommen.

b) Gehe nun B durch und überschreibe A. Schreibe dabei k mal das Element i, wenn an der i-ten Stelle von B ein k steht.

Aufgabe 4. Implementiere folgende Funktion (ins gleiche Modul), die zu einem gegebenen String einen längsten Teilstring findet, der ein Palindrom ist. Speichere diesen Teilstring wieder in s und gib s zurück.

```
1 /* Beschreibung, selber machen */
2 char *str_glsp(char *s);
```

Das Problem ist in polynomieller Laufzeit lösbar und auf die Idee kann man auch kommen.