Aufgabe 1. Mache dich mit `gdb` und `valgrind` vertraut. Sorge mit Hilfe von `valgrind` dafür, dass **jedes** Programm, welches du im Laufe dieses Kurses geschrieben hast, keine Speicherlecks mehr hat.

Aufgabe* 2. Erweitere den Possoinlöser mit Kommandozeilenargumenten. Sorge dafür, dass der User die Anzahl an Stützstellen beim Programmaufruf bestimmen kann. Ferner sollte es die Möglichkeit geben, zwischen verschiedenen rechten Seiten zu wählen, z.Bsp. $g_0(x) = x$ und $g_1(x) = e^{-0.5(x-0.5)^2}$.

Bei dem Kommondozeilenargument "-h" sollte das Programm eine Hilfe ausgeben, in der insbesondere steht, welche Argumente man übergeben kann und was diese bedeuten.

Aufgabe* 3. Schreibe ein Programm, dass Worte aus einer Datei einliest, diese in eine Liste einfügt, anschließend die Liste lexikographisch sortiert und dann die Worte sortiert ausgibt.

Aufgabe* 4. Implementiere den Strassen-Algorithmus zur schnellen Matrixmultiplikation: Für $C = A \cdot B$ mit $A, B, C \in \mathbb{R}^{2^n \times 2^n}$ sei folgende Partitionierung gegeben:

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}; B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}; C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix};$$

wobei alle Untermatrizen gleich groß sind. Definiere 7 neue Matrizen:

$$\begin{aligned} M_1 &:= (A_{1,1} + A_{2,2}) \cdot (B_{1,1} + B_{2,2}) \\ M_2 &:= (A_{2,1} + A_{2,2}) \cdot B_{1,1} \\ M_3 &:= A_{1,1} \cdot (B_{1,2} - B_{2,2}) \\ M_4 &:= A_{2,2} \cdot (B_{2,1} - B_{1,1}) \\ M_5 &:= (A_{1,1} + A_{1,2}) \cdot B_{2,2} \\ M_6 &:= (A_{2,1} - A_{1,1}) \cdot (B_{1,1} + B_{1,2}) \\ M_7 &:= (A_{1,2} - A_{2,2}) \cdot (B_{2,1} + B_{2,2}) \end{aligned}$$

daraus ergibt sich dann für die Lösung C:

$$\begin{aligned} C_{1,1} &= M_1 + M_4 - M_5 + M_7 \\ C_{1,2} &= M_3 + M_5 \\ C_{2,1} &= M_2 + M_4 \\ C_{2,2} &= M_1 - M_2 + M_3 + M_6 \end{aligned}$$

Dabei soll zur Berechnung der M_i natürlich wieder der gleiche Algorithmus zur Matrixmultiplikation verwendet werden. Man spart insgesamt eine Matrixmultiplikation und verringert so den Aufwand von $O(n^3) = O(n^{\log_2(8)})$ auf $O(n^{\log_2(7)})$ also ungefähr $O(n^{2,807})$.