Imperative Programmierung
Aufgabenblatt VIII

1. Zuerst wurde das Projekt aus der Vorlesung rekonstruiert („caesar.c“). Dann wurde nur in der Function „cipher“ die Zeile „omsg[i] = …“ entsprechend angepasst, sodass auch Großbuchstaben berücksichtigt werden.
2. Im Codenknacker („caesar.c“) kann dies auch einfach durch eine weitere Bedingung umgesetzt werden, die bei einem Großbuchstaben, diesen so umrechnet als wäre er klein. Bei kleinen Buchstaben wird „buchstabe – ‘a‘ “ berechnet und bei großen Buchstaben eben „buchstabe – ‘A‘.
3. Um die Erkennung von Großbuchstaben zu vermeiden, habe ich einfach einen globalen Parameter „onlyLower“ erstellt („caesar.c“). Wenn dieser „true“ ist, dann wird einfach jeder Großbuchstabe durch den entsprechenden Kleinbuchstaben ersetzt.
4. Das Programm wurde in „freq.c“ umgesetzt und beinhaltet eigentlich dasselbe wie die Funktion „computeOFreq“ aus „caesar.c“. Nur wird dieses Mal das Zeichen durch „getchar()“ eingelesen und nicht aus einem Array wie in „caesar.c“. Der Text von Moby Dick stammt von dieser URL: https://gist.github.com/StevenClontz/4445774
5. Das Verfahren deutet sehr stark den Bubblesort Algorithmus an, der in der Funktion „sort“ in der Datei „sort.c“ implementiert wurde. Die Funktion „swap“ wurde einfach aus der Vorlesung übernommen. Da der Bubblesort Algorithmus im Grunde genommen auf 2 for Schleifen basiert hat er eine Laufzeit von $O(n^2)$ und gehört damit zu den langsamen Sortierverfahren.
   Als kleines Rahmenprogramm kann zuerst die Länge der Liste und dann alle Zahlen aus der Liste angegeben werden. Danach wird die sortierte Liste ausgegeben.