Aufgabenblatt Häufigkeitsanalyse

Lösen Sie die nachfolgenden Aufgaben und bereiten Sie diese bis zum nächsten Lehrveranstaltungstermin vor.
Ausgangssituation: Sie haben die folgende verschlüsselte Nachricht abgefangen:

```
KSBB GWS RWSGS BOQVFWQVH ZSGSB YCSBBSB, VOPSB GWS RSB GQVZISGGSZ
TISF RWS QOSGOF-QVWTTFS USTIBRSB. KWF VCTTSB, ROGG GWS RWSGS
QVWTTFS BWSAOZG WB GWQVSFVSWHGYFWHWGQVSB OBKSBRIBUSB JSFKSBRSB :).
```

LB-HA 00. (nicht abzugeben)

a) Diskutieren Sie mit welchen einfachen Verschlüsselungsmethoden die obige Nachricht verschlüsselt worden sein könnte. Welche Hinweise geben die Wortstruktur und der verwendete Zeichensatz?

b) Geben Sie an, mit welchen Analyseverfahren der wahrscheinlichste Schlüssel für das hier verwendete Verfahren bestimmt werden kann.

LB-HA 01.

Schreiben Sie ein Programm, dem der Pfad zu einer Textdatei als Argument übergeben wird, das eine Häufigkeitsanalyse der in der Datei vorkommenden Zeichen `'A'` ... `'Z'` durchführt. Das Programm soll diese Zeichen mit zugehöriger **relativer** Häufigkeit in absteigender Reihenfolge auf `std::cout` sortiert ausgeben. Die Häufigkeiten sind in Prozent auf ganze Zahlen kaufmännisch gerundet und zeilenumbruchgetrennt auszugeben, z.B.:

```
E: 17%
N: 10%
I: 8%
[..]
```

Hinweis: Definieren Sie eine geeignete Datenstruktur zur Speicherung der Zeichen und ihrer relativen Häufigkeiten. Verwenden Sie `std::ifstream` (benötigt `<fstream>`) zum Einlesen der Datei und den `>>`-Operator zum Lesen einzelner Zeichen. Verwenden Sie zum Sortieren des Arrays `std::sort` (benötigt `<algorithm>`) mit einer geeigneten eigenen Vergleichsfunktion. Zum Runden bei der Ausgabe über `std::cout` kann mittels `fixed` und `setprecision` (benötigt `<iomanip>`) die Genauigkeit der Ausgabe festgelegt werden.

LB-HA 02.

Schreiben Sie ein Programm, dem ein Eingabedateipfad, ein Ausgabedateipfad und ein Schlüssel (in ebendieser Reihenfolge!) als Argumente übergeben werden, das eine Entschlüsselung der Eingabedatei mit der Caesar-Chiffre in die Ausgabedatei durchführt. Der Schlüssel soll dabei eine Zahl im Intervall $[0; 26[$ sein und angeben, um wie viele Zeichen bei der Verschlüsselung verschoben wird.

Andere Zeichen als `'A'` ... `'Z'` sollen **nicht** entschlüsselt, sondern unverändert in die Ausgabedatei übernommen werden.
Hinweis: Verwenden Sie neben den in Beispiel LB-HA 01. angewendeten Dateieingabefunktionen zusätzlich `std::ofstream` *(benötigt* `<fstream>`*) zum Ausgeben der Datei und den* `<<`*-Operator zum Schreiben einzelner Zeichen. Verwenden Sie zum Umwandeln des Schlüssels vom Argument (*`const char* const`*) in eine Zahl (*`int`*) die Funktion* `atoi` *(benötigt* `<cstdlib>`*). Stellen Sie vor der Entschlüsselung sicher, dass der oben angegebene Wertebereich des Schlüssels eingehalten wird.*