Aufgabenblatt Häufigkeitsanalyse

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

```
GVLX JGO UGOJC XRARIGMYR VVQOE IYVLXVL, RRZOE QSV BOE QMYJEVQCVJ
PLCB UGO MGQVLOIC-MYGPWPO XCPLLNVL. GZP RFDPVL, NRQC JGO UGOJC
MYGPWPO EGODYVJ GX JGMYCBYCSKQUIGDZQMYCX RLGVLNLLQVL FVPGVLNVL :).

ED BSV FKVSPZEUVGDJYXRJIJC CKYBB XE MCBVGXWYMYCX, WMVXR NVP OIQDV
YLJYDQ YEJ DBRLJ BYPBYC UGO MCBNYXUJEEE:

KCQ QICQFP CRKCR CSECC DMBXCXJ YEJ SXISRZEOE RBRCEDCX VPGRARKC,
PRLN VP CZAR ZL CVGXVK LVRD QS OZLOD SXXCRVSOICX LLQVXSVDOI
TOIUKEBOCR. OI JKX YEW QOZLOD NKEXOIYBKGQ YYBKCX ISOTIOE SXU QKY,
UOEL OI BOE IYGD OZL GVLSX FYS, QOZLOE EONMOCZDVL, LIYEECX, MMX
SMQVLPFCBDGQVL FVPCKCSWSXXCX XCDVGVKCX SYETF, KLD NVQCVL RFCRV
QSTF NZC LVRDUCMBC, JLK QRCXQJSTFOE LSVBOIEVVGDVL LVPOZR, URSW
EMMY CBYYVKCX BMXERO. JCSEC FZCVVL, SD TOIEVVGMY XE JCSECW
JMXJRSXCX LKPRLQ BJKVEVZAR USOELOE ZOZLO WJSDKOIROE GRD FSCDVFQ
FFP NVL KLEOE.
```

Es sei bekannt, dass die Nachricht in deutscher Sprache verfasst wurde.

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.

Platzhalter für Lösung:

Ergebnis der _____-Analyse:

Weitere Analyse (_____):

Die Schlüssellänge beträgt _____.

LB-HA 01. (*Code::Blocks*-Template *Small cryptographic project*)

Zur Bestimmung des wahrscheinlichsten Schlüssels soll zunächst die obige Nachricht in drei Teile zerlegt werden – einen pro Schlüsselposition. 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'` für jede Schlüsselposition durchführt. Das Programm soll diese Zeichen mit zugehöriger **relativer** Häufigkeit (bezogen **nur** auf die zu analysierenden Zeichen pro Schlüsselposition) in absteigender Reihenfolge und pro Schlüsselposition spaltenweise auf `std::cout` **stabil** sortiert ausgeben. Die Häufigkeiten sind pro Zeichen in Prozent auf ganze Zahlen kaufmännisch gerundet und zeilenumbruchgetrennt **exakt** (d.h. inklusive Leerzeichen und Trennzeichen) wie folgt auszugeben (die Zahlenwerte sind nur Beispiele):

```
E: 17% ; F: 17% ; D: 16%
N: 10% ; O: 9% ; M: 10%
I: 8% ; J: 8% ; H: 10%
[..]
```

Anmerkung: `[..]` ist als Platzhalter für die verbleibenden Zeilen zu verstehen und **nicht** Teil der Ausgabe. Die erste Spalte entspricht der Häufigkeitsverteilung von Buchstaben in der deutschen Sprache (ohne Verschlüsselung).
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 kann* `round` *(benötigt* `<cmath>`*) verwendet werden. Die Genauigkeit der Ausgabe über* `std::cout` *kann mittels* `fixed` *und* `setprecision` *(benötigt* `<iomanip>`*) festgelegt werden. Die Nachricht soll für die Häufigkeitsanalyse nicht in drei Dateien aufgeteilt werden, sondern die Zugehörigkeit zur Schlüsselposition soll im Programm anhand der Zeichenposition in der Datei ermittelt werden.*

LB-HA 02. (*Code::Blocks*-Template *Small cryptographic project*)

Die Caesar-Chiffre ist wie folgt definiert: Ein Klartextzeichen m (als Zahl von 0 bis 25 dargestellt) wird mit dem Schlüsselzeichen k (als Zahl von 0 bis 25 dargestellt) durch Verschiebung zu einem Zeichen c verschlüsselt.

$$c = (m + k) \mod 26$$

Das Zeichen `'A'` entspricht dabei der Zahl 0, das Zeichen `'B'` der Zahl 1 usw. Analog zur Verschlüsselung kann das Zeichen c durch entsprechendes Rückverschieben um k zu m entschlüsselt werden.

$$m = (c - k) \mod 26$$

Die Vigenère-Chiffre entspricht einer zyklischen positionsweisen Anwendung der Caesar-Chiffre mit einem l Zeichen langen Schlüssel:

$$c_i = (m_i + k_{i \mod l}) \mod 26$$

Das bedeutet, dass beispielsweise bei einer Schlüssellänge von drei Zeichen das vierte Klartextzeichen wiederum mit dem ersten Schlüsselzeichen verschlüsselt wird.

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 Vigenère-Chiffre in die Ausgabedatei durchführt. Andere Zeichen als `'A'`...`'Z'` sollen **nicht** entschlüsselt, sondern unverändert in die Ausgabedatei übernommen werden. Prüfen Sie die Korrektheit Ihres Programmes mit der oben angegebenen Nachricht und dem Schlüssel, den Sie wie folgt ermitteln können:
Verwenden Sie das Ergebnis der Häufigkeitsanalyse aus Beispiel LB-HA 01., um zeichenweise den wahrscheinlichsten Schlüssel zu berechnen. Jede Spalte entspricht dabei einer Schlüsselposition. Ermitteln Sie den Versatz zwischen der Häufigkeitsverteilung der Buchstaben der deutschen Sprache (vgl. oben) und der vorgefundenen Häufigkeitsverteilung. Dieser Versatz entspricht dabei dem jeweiligen Schlüsselzeichen. In der Beispielausgabe zu LB-HA 01. oben würde der Schlüssel beispielsweise `'ABZ'` lauten, da der Versatz in der ersten Spalte 0 (entspricht `'A'`), der der zweiten Spalte 1 (entspricht `'B'`) und der der dritten Spalte $-1 = 25 \mod 26$ entspricht `'Z'`) beträgt.
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. Stellen Sie vor der Entschlüsselung sicher, dass der Schlüssel nur Zeichen im angegebenen Wertebereich enthält.*