CS 371: Introduction to Computer Networking
Spring 2023, University of Kentucky

Assigned: Apr 13, 2023
Due: Apr 22, 2023

CS 371 Programming Assignment 3

Preliminaries:

1. **Verify the access to your assigned Linux VM.** See instruction from PA1.
2. Download the support code from Canvas, a compressed file named `PA3support.tgz`. Please decompress it and verify that the files are structured as follows:

```
PA3support/
└─Cryptanalysis/
   ├─ciphertext.txt
   ├─hamlet.txt
   └─merchantofvenice.txt
```

3. **Build and test your programs in Linux VM**. See instruction from PA1. In case you need to install a new programming language package (e.g., Python), feel free to do so.

Problem 1: Cryptanalysis (70 pts).

The objective of this problem is to explore the security issues of **monoalphabetic substitution ciphers** (textbook Ch. 8.4.3). You are expected to design, implement, and evaluate a frequency-based cryptanalysis tool that can decipher ciphertext generated by a monoalphabetic substitution cipher.

The first task involves writing a program `freqAnalyze.*` in any programming language (e.g., `freqAnalyze.py`, `freqAnalyze.cpp`, etc.) to analyze the letter frequencies when given an input of reference text. Use the following guidelines when designing `freqAnalyze.*`:

- Ignore case (upper and lower count the same);
- It should sort the counts of single letters, pairs of letters (bigrams), and triples of letters (trigrams), then print the non-zero values in decreasing order. For single letters, print all non-zero values; for bigrams and trigrams, print the first 30 non-zero values.

The second task is to use `freqAnalyze.*` to produce frequency analysis results for several given reference texts. Then, based on the results, try deciphering a given ciphertext, which was produced by a monoalphabetic substitution cipher. The deciphering process requires using `freqAnalyze.*` on the ciphertext too and creating a letter-to-letter mapping. A correct mapping will help you restore the original plaintext. When finished, write a `decipher.*` program that takes the ciphertext as input and outputs the deciphered plaintext.

(*Hint: the creation of the mapping may involve human effort and be done gradually. When you gradually update the mapping, the partially deciphered text would also make partial sense to you.)

Files to turn in: `freqAnalyze.*`, `decipher.*`, `documentation.txt`, and/or any other files.

Scoring:

- **Frequency analysis (20 pts):** Your `freqAnalyze.*` program should compile/run successfully (based on your instruction in `documentation.txt`) without errors or warnings in your Linux VM. It should output in the command line the correct frequency analysis results for the two given reference texts `hamlet.txt` and `merchantofvenice.txt` respectively.
- **Deciphering (20 pts):** Your `decipher.*` program should compile/run successfully (based on your instruction in `documentation.txt`) without errors or warnings in your Linux VM. It should output in the command line the correct deciphered plaintext for the given `ciphertext.txt`.
- **Documentation (20 pts):** The `documentation.txt` should include:
  1. A brief discussion on how you designed and implemented `freqAnalyze.*` and `decipher.*`;
  2. How to compile/run your `freqAnalyze.*` and `decipher.*` programs;
  3. Your self-evaluation results and any interesting observation for the frequency analysis and deciphering tasks.
- **Code readability and clarity (10 pts).**

Write-up (10 pts): Create a `writeup.txt` specifying the following:

1. Name and linkblue ID of each member in your group.
2. A brief statement on the contribution of each group member in finishing the assignment.

Canvas Submission Instruction: submit a file named `PA3-groupnumber.tgz` (replace `groupnumber` with your group number). This `.tgz` file when uncompressed should contain the following structure:

```
PA3-groupnumber/
├── Cryptanalysis/
│   ├── freqAnalyze.*
│   ├── decipher.*
│   ├── documentation.txt
│   └── (any_new_files)
└── writeup.txt
```