CS 371 Programming Assignment 3

Preliminaries:

1. Download the support files 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
```

2. **Build and test your programs in Linux VM**. See instruction from PA1. In case you are using Python, `python3` should already be installed in the VM.

Problem 1: Cryptanalysis (60 pts).

The objective of this problem is to explore **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.

Task 1: 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 (one-grams), two-letters (bi-grams), and three-letters (tri-grams), then print the non-zero counts in decreasing order.
- For single letters, print all non-zero values; for bigrams and trigrams, print the 10 highest non-zero values.

Task 2: 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.txt`, which was produced by a monoalphabetic substitution cipher. The deciphering process requires using `freqAnalyze.*` on the ciphertext to have a similar result of ranked frequency analysis as the first step. Then, by comparing the results from those in **Task 1**, you can create a one-on-one letter mapping (i.e., codebook). A correct mapping will help you restore the original 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.)

When finished, write a `decipher.*` program that encodes the mapping which can take a ciphertext as input and output the plaintext in the command line.

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

Scoring:

- **Frequency analysis (15 pts):**
  - Your `freqAnalyze.*` program, when taking either of the references `hamlet.txt` and `merchantofvenice.txt`, should compile/run successfully (based on your instruction in `writeup.txt`) without errors or warnings in your Linux VM.
  - The above step should output the correct frequency analysis results (see **Task 1**) for the two given reference texts respectively in the command line. Consider using this output format:

Index	OneGrams	Counts	TwoGrams	Counts	TriGrams	Counts
1	e	15846	th	3938	the	1954
2	t	12450	he	2979	and	1138
3	o	11450	er	2079	you	852
4	a	10249	an	1981	hat	697

    (this is only a reference for format; counts may be different in the actual tests. You need to show all the 26 entries for one-grams and the top 10 entries for bi-grams and tri-grams, per **Task 2**.)

- **Deciphering (15 pts):**
  - Your `decipher.*` program, when taking input `ciphertext.txt`, should compile/run successfully (based on your instruction in `writeup.txt`) without errors or warnings in your Linux VM.
  - The above step should output the one-on-one letter mapping (i.e., codebook) and the deciphered plaintext in the command line (see **Task 2**).
- **Documentation (10 pts):** Create a `documentation.txt` to include:
  1. A step-by-step description of how you designed and implemented your solution; this should include the frequency analysis results, one-on-one letter mapping, and the deciphered plaintext obtained by yourself.
  2. A brief description of how to compile/run your programs.
  3. Any interesting observation and/or explanation of the results.
- **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 compressed file named `PA3-Group##.tgz` (replace `##` with yours, use two digits), which when uncompressed should contain the following:

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