SCHOOL OF COMPUTING (SOC)

Diploma in Applied AI and Analytics

ST1507 DATA STRUCTURES AND ALGORITHMS (AI)

2023/2024 SEMESTER 2
ASSIGNMENT ONE (CA1)
~ Caesar Cipher Encrypted Message Analyzer ~

Objective of Assignment

To practice what you have learnt in the module on data structures, algorithms, and object-oriented programming by developing a Caesar Cipher Encrypted Message Analyzer Application.

Instructions and Guidelines:

1. This is an individual assignment, and it accounts to **30%** of your final grade.
2. The submission date is **Friday 24 November 5:00 pm**.
3. The development will be carried out in Python using Anaconda. You should only make use of those libraries that already ship with Anaconda Jupyter. You should not make use of the collection library.
4. The interviews will be conducted during the DSAA lessons in week 7. You are expected to explain your code and program logic. Take note that the interview is compulsory.
5. No marks will be given if the work is copied, or you have allowed others to copy your work.
6. **50% of marks** will be deducted for submission of assignment within **ONE** calendar day after the deadline. **No marks shall** be awarded for assignments submitted **more than one day** after the deadline.

Warning: Plagiarism means passing off as one's own the ideas, works, writings, etc., which belong to another person. In accordance with this definition, you are committing plagiarism if you copy the work of another person and turning it in as your own, even if you would have the permission of that person.
Plagiarism is a serious offence, and if you are found to have committed, aided, and/or abetted the offence of plagiarism, disciplinary action will be taken against you. If you are guilty of plagiarism, you may fail all modules in the semester, or even be liable for expulsion.

Overview of the system

Your job is to implement a *Caesar Cipher(*) Encrypted Message Analyzer* Application The application will serve the following three purposes:

1. It will allow a user to encrypt and decrypt messages typed on the screen or read from files.
2. It will allow a user to analyze letter frequencies distributions and use that to infer an (unknown) Caesar cipher key from an encrypted file.
3. It will allow the user to analyze and sort multiple Caesar encrypted files that are stored in a folder.

(*) A Caesar Cipher is a special instance of a substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down (or up) the alphabet. Take note, you can read more on Caesar Cipher at Wikipedia.
https://en.wikipedia.org/wiki/Caesar_cipher

An example of Caesar Cipher encryption:

Next is an example demonstrates how a message (plaintext) can be encrypted with a Caesar Cipher key of -1 (Take note, that a negative key implies a left shift).

Example using a Caesar Cipher key of -1 (left shift):

Plaintext:
```
The first four letters of our alphabet are, ABCD.
```
Ciphertext:
```
Sgd ehqrs entq kdssdqr ne ntq zkogzads zqd, ZABC.
```

We may decrypt this message by using a Ceasar Cipher with the same shift, but now in the opposite direction. So, in this case by using a key of 1 (Take note, that a positive key implies a right shift)

Example using a Caesar Cipher key of 1 (right shift):

Plaintext:
```
Sgd ehqrs entq kdssdqr ne ntq zkogzads zqd, ZABC.
```
Ciphertext:
```
The first four letters of our alphabet are, ABCD.
```

Next example demonstrates how you may use either a negative key (left shift) or a positive key (right shift) as to obtain the same Caesar Cipher encryption.

For instance, using a cipher key of -3 (left shift) on the below plain text produce the following ciphertext.

Plaintext:
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
Ciphertext:
QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

We may obtain the same encryption by using a cipher key of 23 (right shift).

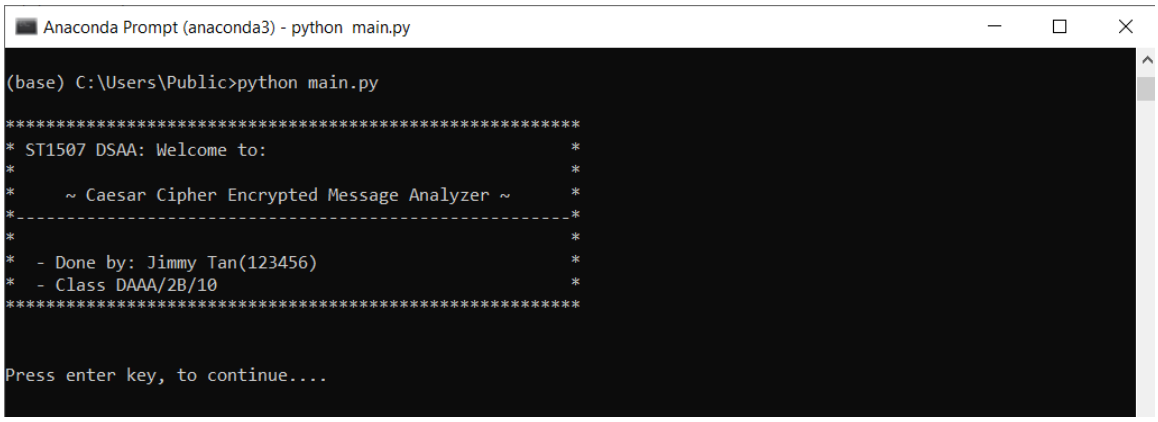
Plaintext:
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
Ciphertext:
QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

Starting the application

Your application must be able to start from the Anaconda Command prompt by typing:

`python main.py`.

It will then display a title bar with essential information, next it waits for the user to press the enter key before the application continues.



- You are required to follow the above format.
- Please, ensure that you display your name, student ID and class in the format as is shown in the example.

Selection menu

When the application continues, the user will then be presented with a menu as shown below. The menu allows the user to choose from 8 options.

```
Please select your choice: (1,2,3,4,5,6,7,8)
        1. Encrypt/Decrypt Message
        2. Encrypt/Decrypt File
        3. Analyze letter frequency distribution
        4. Infer caesar cipher key from  file
        5. Analyze, and sort encrypted files
        6. Extra Option One
        7. Extra Option Two
        8. Exit
Enter choice:
```

- You are required to follow the above format for the menu display.
- Option 6 and 7 are reserved as extra options that you will need to design and implement yourself.
- The user will be able to repeatedly select the various options from the menu.
- The application will terminate once the user selects option 8 Exit (displaying an Exit message as shown below).

```
Press enter key, to continue....

Please select your choice: (1,2,3,4,5,6,7,8)
        1. Encrypt/Decrypt Message
        2. Encrypt/Decrypt File
        3. Analyze letter frequency distribution
        4. Infer caesar cipher key from  file
        5. Analyze, and sort encrypted files
        6. Extra Option One
        7. Extra Option Two
        8. Exit
Enter choice: 8

Bye, thanks for using ST1507 DSAA: Caesar Cipher Encrypted Message Analyzer
```

Encrypting/Decrypting messages

The user may select option '1' to encrypt or decrypt a message typed in the screen. The user will be prompted to choose either 'E' for encryption or 'D' for decryption. Thereafter the user will be required to type in a single line of plain text, followed by a request to type in the cipher key. For instance, in the below screen we are performing an encryption with a cipher key of -3.

```
Please select your choice: (1,2,3,4,5,6,7,8)
        1. Encrypt/Decrypt Message
        2. Encrypt/Decrypt File
        3. Analyze letter frequency distribution
        4. Infer caesar cipher key from  file
        5. Analyze, and sort encrypted files
        6. Extra Option One
        7. Extra Option Two
        8. Exit
Enter choice: 1

Enter "E" for Encrypt or "D" for Decript: E

Please type text you want to encrypt:
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Enter the cipher key: -3

Plaintext:       THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
Ciphertext:      QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

Press enter key, to continue....
```

Below shows how we may decrypt the previous message with the same cipher key of -3.

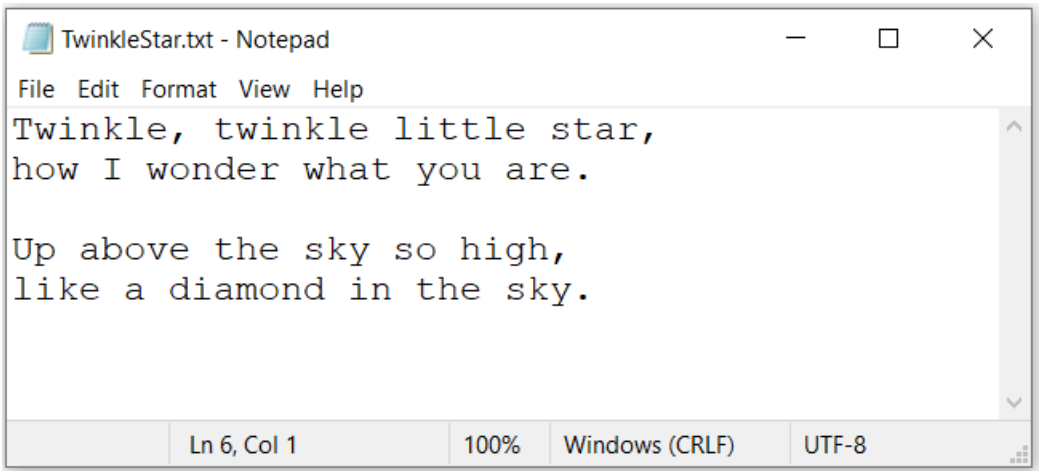
```
Enter choice: 1

Enter "E" for Encrypt or "D" for Decript: D

Please type text you want to decrypt:
QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

Enter the cipher key: -3

Ciphertext:      QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD
Plaintext:       THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Press enter key, to continue....
```

Encrypting/decrypting files

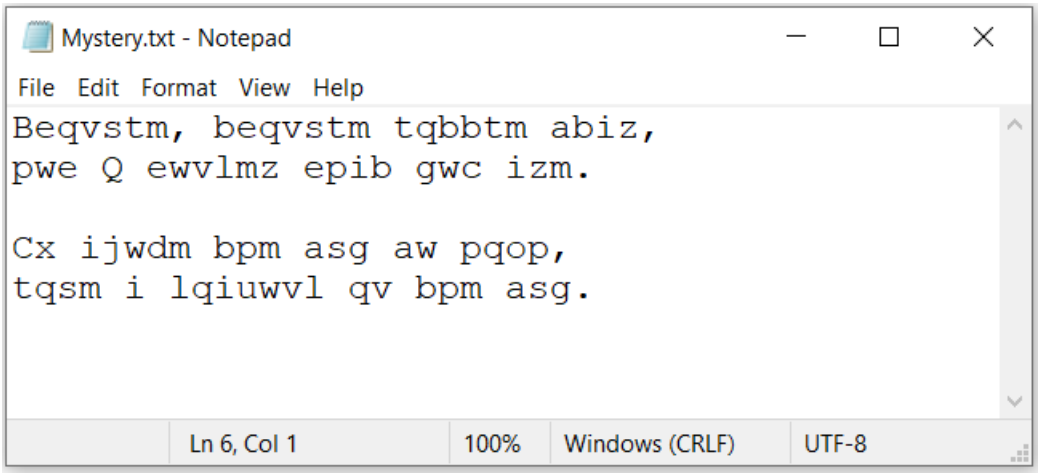
The user may select option ‘2’ to encrypt or decrypt a text from a file. The user will be prompted to choose either ‘E’ for encryption or ‘D’ for decryption. Thereafter the user will be required to type the name of both an input file (as to read the plain text from) and an output file (as to write the encrypted text too), followed by a request to type in the cipher key.

```
Please select your choice: (1,2,3,4,5,6,7,8)
        1. Encrypt/Decrypt Message
        2. Encrypt/Decrypt File
        3. Analyze letter frequency distribution
        4. Infer caesar cipher key from  file
        5. Analyze, and sort encrypted files
        6. Extra Option One
        7. Extra Option Two
        8. Exit
Enter choice: 2

Enter "E" for Encrypt or "D" for Decript: E

Please enter the file your want to encrypt: TwinkleStar.txt

Enter the cipher key: 8

Please enter a output file: Mystery.txt

Press enter key, to continue....
```

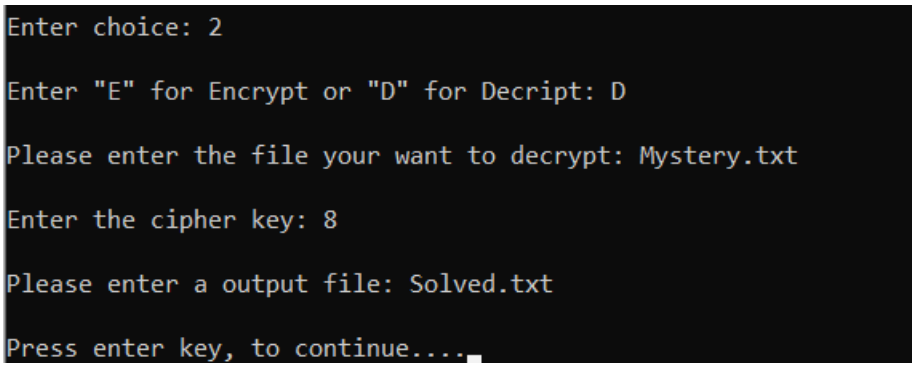
In this example we are doing an encryption with a cipher key of 8. The plaintext is as follows:



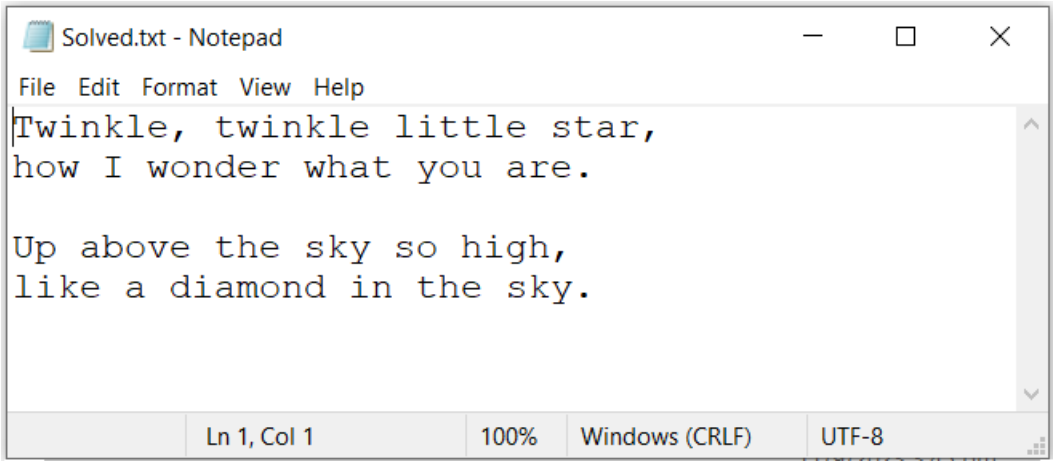
The encrypted text will end up as follows:



We may decrypt the file by using the same key of 8, as is shown below.

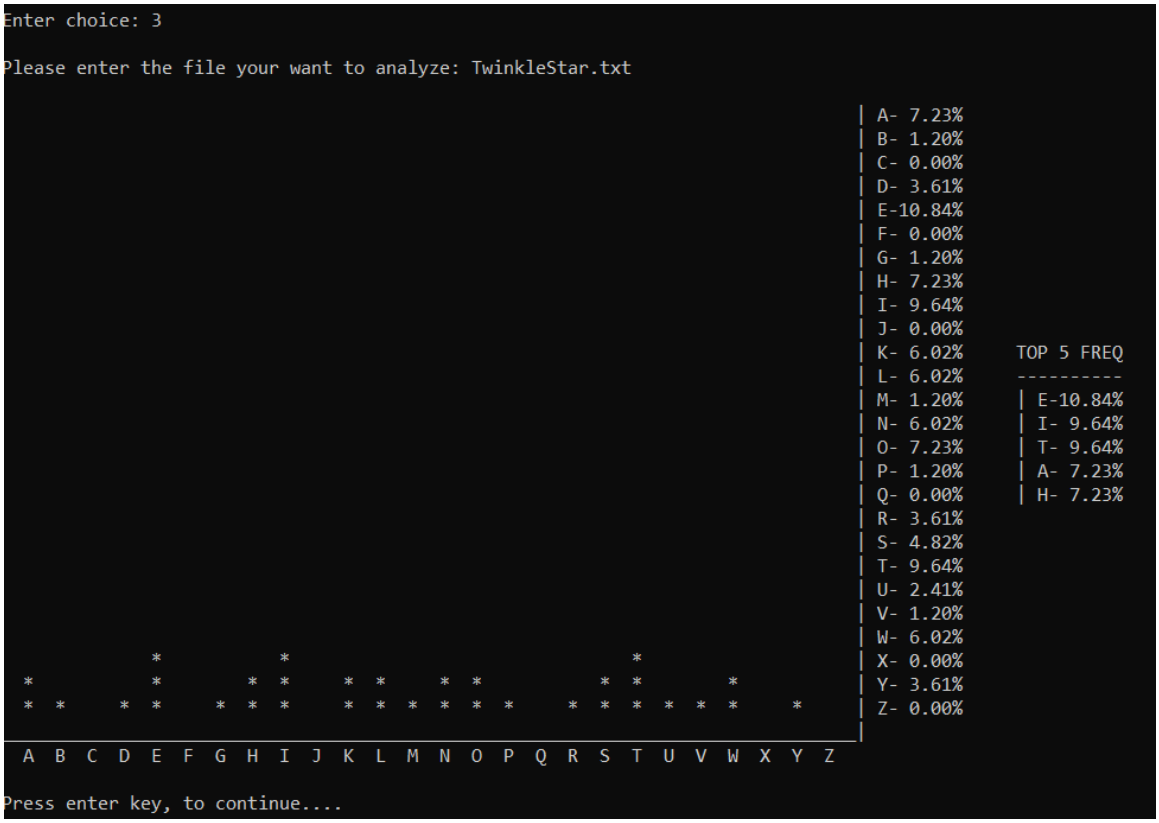


It would give us back the original plaintext.

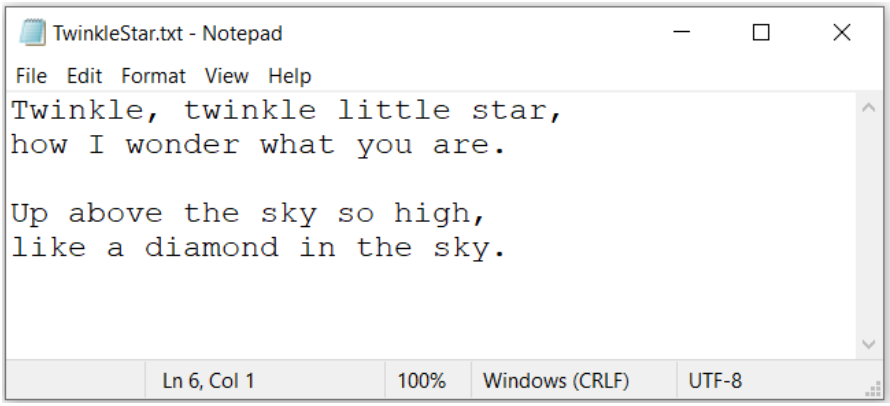


Analyze letter frequency distribution

A user can perform a letter frequency distribution analyzes, by selecting option '3'. After entering the filename, the letter frequencies will be plotted as a bar graph (as shown below). At the right-hand side, it will display the frequency of each letter in the alphabet (rounded to two decimals). Also, the top 5 frequencies, sorted by highest frequency will be displayed. Take note that when multiple letters have the same frequencies they will be sorted by ascending in alphabetical order when listing the top 5 frequencies.



The text that was used for the above displayed letter frequency distribution analysis was as follows:

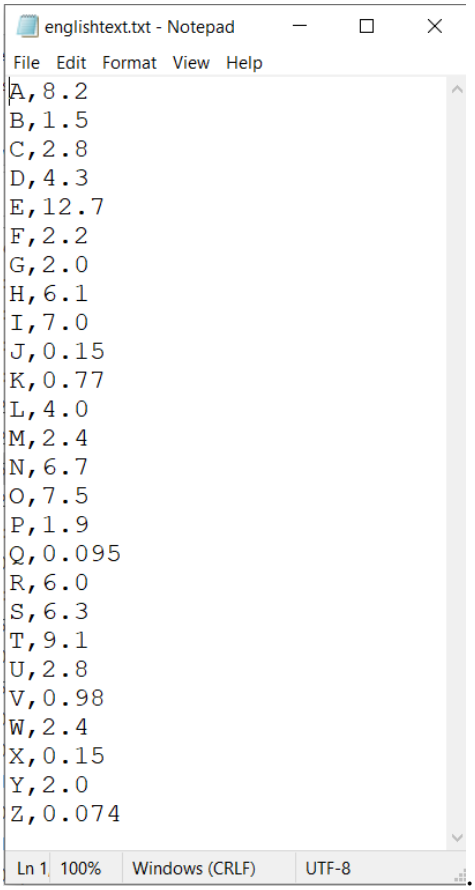


Infer Caesar Cipher Key from file

A Caesar Cypher encryption is easy to be break by analyzing and comparing the letter frequency distribution from the encrypted text with that from the frequency distribution for letters in English text in general (based of averages of large amounts of English text). In crypto-analysis, frequency analysis is a technique to break simple ciphers such as Caesar cipher for situations whereby the cipher key is unknown to you. The letter frequency distribution in English text has a distinct pattern, that is merely shifted when you do a Caesar encryption on a text, and hence this would allow you to infer the cipher key with some effort. You may read more on this technique at this Wikipedia site.

https://en.wikipedia.org/wiki/Frequency_analysis

The frequencies for letters in English text (average occurrences of letters in English text) is given to you on Brightspace as the file 'Englishtext.txt' (*). Take note each letter is followed by the percentage of the average occurrence of that letter in English text. Question can you spot the top 5 letters?

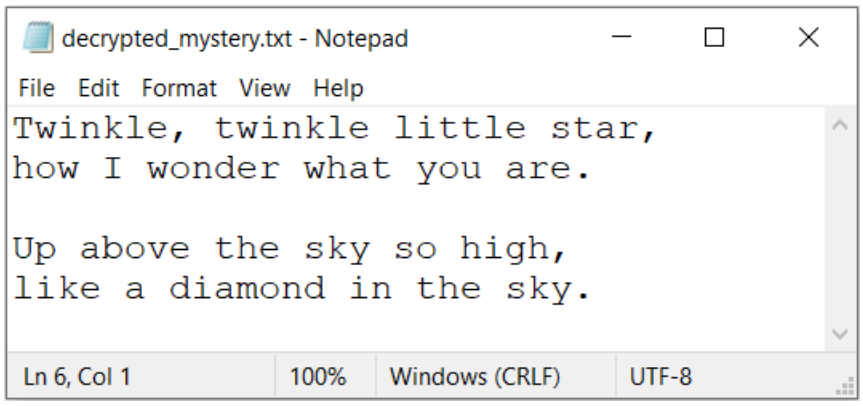


(*) Source Wikipedia:
https://en.wikipedia.org/wiki/Letter_frequency

By selecting option ‘4’ a user can perform a letter frequency distribution analysis as to infer a cipher key from a encrypted text. After entering the file to be analyzed, the user will be prompted to enter the file that will be used as a reference. The application will then infer the Caesar key through a letter frequency distribution analysis. Once that is done the application returns the inferred Caesar key. The user can then, optionally, use that key to decrypt the encrypted file.

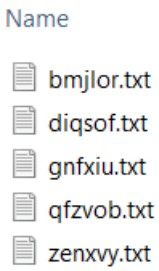
```
Press enter key, to continue....

Please select your choice: (1,2,3,4,5,6,7,8)
        1. Encrypt/Decrypt Message
        2. Encrypt/Decrypt File
        3. Analyze letter frequency distribution
        4. Infer caesar cipher key from  file
        5. Analyze, and sort encrypted files
        6. Extra Option One
        7. Extra Option Two
        8. Exit
Enter choice: 4

Please enter the file to analyze: Mystery.txt

Please enter the reference frequencies file: englishtext.txt
The inferred caesar cipher key is: 8
Would you want to decrypt this file using this key? y/n: y

Please enter a output file: decrypted_mystery.txt

Press enter key, to continue....
```

In this case using the inferred key of 8, the text will be decrypted to the original plain text.



Analyze and sort encrypted files

The user can use option '5' to do batch processing of multiple encrypted files that are stored in a folder. For instance, a folder may contain the following 5 encrypted files.



- Take note, this folder is assumed to be in the same directory where your main.py file is located.

The user will be prompted to enter a folder name, containing the encrypted files.

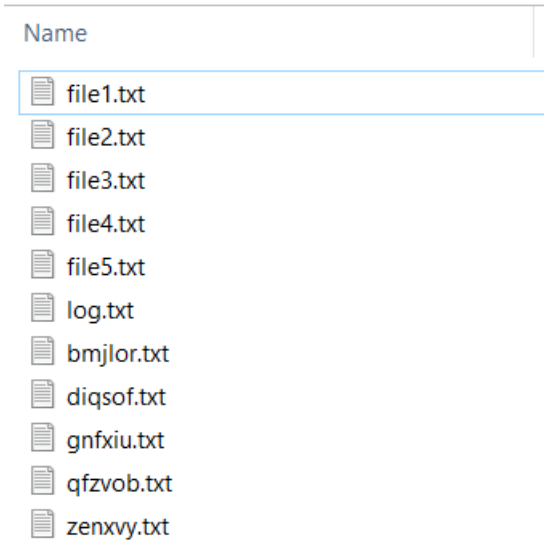
```
Press enter key, to continue....

Please select your choice: (1,2,3,4,5,6,7,8)
        1. Encrypt/Decrypt Message
        2. Encrypt/Decrypt File
        3. Analyze letter frequency distribution
        4. Infer caesar cipher key from  file
        5. Analyze, and sort encrypted files
        6. Extra Option One
        7. Extra Option Two
        8. Exit
Enter choice: 5

Please enter the folder name: CASE01
Decrypting: qfzvob.txt with key: 2 as: file1.txt

Decrypting: gnfxiu.txt with key: 5 as: file2.txt

Decrypting: zenxvy.txt with key: 19 as: file3.txt

Decrypting: diqsof.txt with key: 21 as: file4.txt

Decrypting: bmjlor.txt with key: 25 as: file5.txt


Press enter key, to continue....
```

All the files in the folder will then be analyzed as to infer the encryption keys and then decrypted with those inferred keys. The resulting output files will be named after the ascending order of the inferred keys, as file1.txt, file2.txt, file3.txt …. etc.

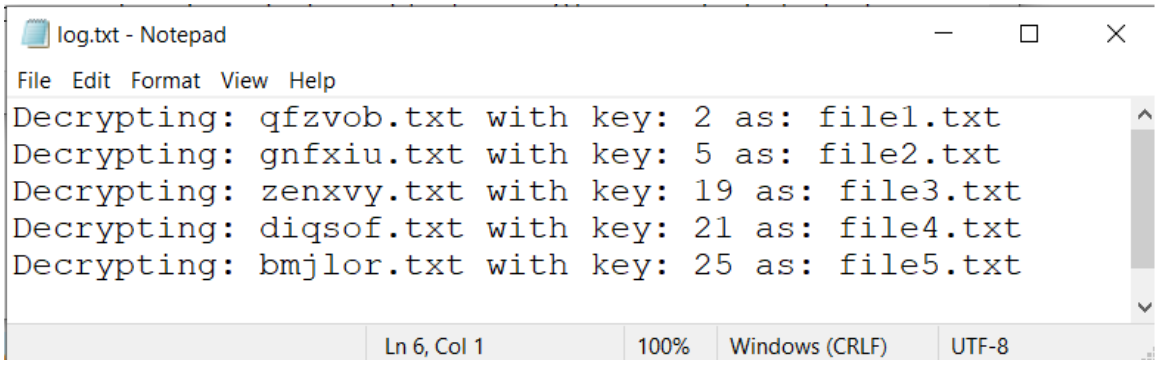
- Take note you may assume that each encrypted file in the folder was encrypted with a unique key.

- Also take note, all keys inferred will be returned as right shift keys, so they will all be positive numbers.

After the batch processing of files is completed, the folder will contain the original encrypted files, the decrypted files, and a log file (as show in below example_.



The log file will indicate for each decrypted file, from what file it was decrypted, and with what key. Take note that the filenames for the decrypted files will reflect the order of the keys in ascending order. Below is an example of what the log file could look like as for this particular case.



Exiting the application

The user can repeatedly select Options 1,2,….,7. Option 8 is to exit the program. Once the user exits the application you should display the message shown below.

```
Please select your choice: (1,2,3,4,5,6,7,8)
        1. Encrypt/Decrypt Message
        2. Encrypt/Decrypt File
        3. Analyze letter frequency distribution
        4. Infer caesar cipher key from  file
        5. Analyze, and sort encrypted files
        6. Extra Option One
        7. Extra Option Two
        8. Exit
Enter choice: 8

Bye, thanks for using ST1507 DSAA: Caesar Cipher Encrypted Message Analyzer
```

Basic requirements:

- You are required to design and write the Python application using an object-oriented programing approach (OOP). You should thereby leverage on your knowledge of encapsulation, function/operator overloading, polymorphism, inheritance etc.

- You may make use of Python's already built in data structures, such as list, tuple, dictionary etc., however you should refrain from using the classes from the collection library. Instead, you are required to write you own classes to support the various data structures that you may need (for instance for sorting you may need a class *SortedList*). Of course, you may refer to the lecture slides and lab tasks and expand further on those classes that we had previously developed and discussed in tutorials and lab sessions.

- All the classes that you develop must be placed in separate python files.

- Pay attention to user input validation. Your application should not crash if a user types in the wrong input. Instead, when a user enters wrong input, you should notify the user, and allow him/her to enter again.

Deliverables

Your deliverables must include all the python files (.py files).

- Make sure that the code you submit is complete, and that it can run from the Anaconda prompt as:

  ```
  python main.py
  ```

- Make sure your code is properly commented, and that for each python file you write on top in comments, your name, student ID and class.

- Include some examples of encrypted, and decrypted text files for processing that you may have been using for testing your application (take note assessors may test your application with their own files).

- You must include in your submission a duly filled in and signed Academic Integrity Declaration form.

Submission requirements

- Submit all the deliverables (Source Code, test files, Academic Integrity Declaration form) in the designated BrightSpace drop box before the submission deadline.

- You must submit it as one Zipped folder (RAR will not be accepted, only zip).

  Label your submission as:

  CA1_Name_StudentID_Class.zip

  For example: *CA1_JimmyTan_12345_2B10.zip*

Assessment Criteria

The assignment will be assessed based on the following criteria:

Assessment criteria	Marks awarded
User Interface and File IO - An interactive user interface that follows the required format, as shown in screenshots. - Proper user input validation (handles wrong user inputs correctly). - Reading and writing of files is functional.	Max 20
Functionality of the application: - Able to use Caesar Cipher to encrypt and decrypt messages from screen and files. - Able to perform, and display letter frequencies distributions analyzes according to the required format, and to use that to infer keys from encrypted files. - Support batch processing of encrypted files as to analyze, decrypt and sort the resulting files.	Max 30
Programming techniques, efficiency, robustness, and readability of code: - Usage of classes and OOP technology. - Usage of data structures (python build in data structures as well as those data structures that you made yourself). - Functions and programming constructs. - Code is properly commented. - Application if free of crashes.	Max 30
Two Extra Features: - Features will be judged according to technical sophistication and useability of feature.	Max 10
Interview/Demonstration: - Ability to demonstrate and explain the application and the code clearly. - Q&A **(*) Multiplier on total score may be applied if student demonstrates poor understanding of code.**	Max 10
Grand Total	**100**

-- The End --