Caesar and Vigenère Ciphers

COMP 273 Assignment 3 - Winter 2021, Prof. Kry

LI BRXWK, WKURXJKRXW DOO KLVWRUB, KDG KDG D FKDPSLRQ WR VWDQG XS IRU LW; WR VKRZ D GRXEWLQJ ZRUOG WKDW D FKLOG FDQ WKLQN; DQG, SRVVLEOB, GR LW SUDFWLFDOOB; BRX ZRXOGQ'W FRQVWDQWOB UXQ DFURVV IRONV WRGDB ZKR FODLP WKDW "D FKLOG GRQ'W NQRZ DQBWKLQJ." D FKLOG'V EUDLQ VWDUWV IXQFWLRQLQJ DW ELUWK; DQG KDV, DPRQJVW LWV PDQB LQIDQW FRQYROXWLRQV, WKRXVDQGV RI GRUPDQW DWRPV, LQWR ZKLFK JRG KDV SXW D PBVWLF SRVVLELOLWB IRU QRWLFLQJ DQ DGXOW'V DFW, DQG ILJXULQJ RXW LWV SXUSRUW. XS WR DERXW LWV SULPDUB VFKRRO GDBV D FKLOG WKLQNV, QDWXUDOOB, RQOB RI SODB. EXW PDQB D IRUP RI SODB FRQWDLQV GLVFLSOLQDUB IDFWRUV. "BRX FDQ'W GR WKLV," RU "WKDW SXWV BRX RXW," VKRZV D FKLOG WKDW LW PXVW WKLQN, SUDFWLFDOOB RU IDLO. QRZ, LI, WKURXJKRXW FKLOGKRRG, D EUDLQ KDV QR RSSRVLWLRQ, LW LV SODLQ WKDW LW ZLOO DWWDLQ D SRVLWLRQ RI "VWDWXV TXR," DV ZLWK RXU RUGLQDUB DQLPDOV. PDQ NQRZV QRW ZKB D FRZ, GRJ RU OLRQ ZDV QRW ERUQ ZLWK D EUDLQ RQ D SDU ZLWK RXUV; ZKB VXFK DQLPDOV FDQQRW DGG, VXEWUDFW, RU REWDLQ IURP ERRNV DQG VFKRROLQJ, WKDW SDUDPRXQW SRVLWLRQ ZKLFK PDQ KROGV WRGDB. EXW D KXPDQ EUDLQ LV QRW LQ WKDW FODVV. FRQVWDQWOB WKUREELQJ DQG SXOVDWLQJ, LW UDSLGOB IRUPV RSLQLRQV; DWWDLQLQJ DQ DELOLWB RI LWV RZQ; D IDFW ZKLFK LV VWDUWOLQJOB VKRZQ EB DQ RFFDVLRQDO FKLOG "SURGLJB" LQ PXVLF RU VFKRRO ZRUN. DQG DV, ZLWK RXU GXPE DQLPDOV, D FKLOG'V LQDELOLWB FRQYLQFLQJOB WR LPSDUW LWV WKRXJKWV WR XV, VKRXOG QRW FODVV LW DV LJQRUDQW.

Submission instructions

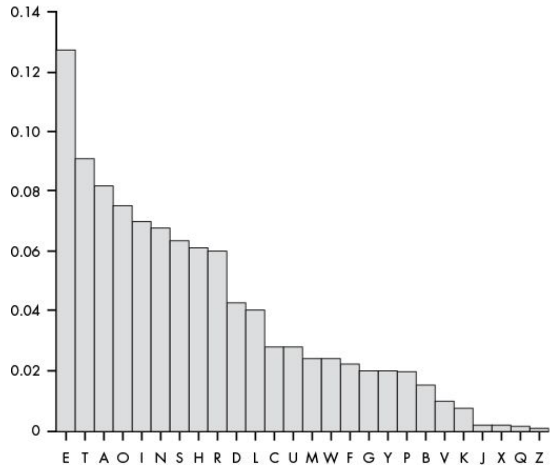
All work must be your own, and must be submitted by MyCourses. **Include your name and student number in a comment at the top of your source file**. Also use comments at the top of your code to provide any other information you would otherwise include in a README file. Submit only one file, `cipher.asm`, i.e., you will need to rename the provided file! Do not use a zip archive. **Check your submission** by downloading your submission from the server and checking that it was correctly submitted. You will not receive marks for work that is incorrectly submitted.

1 Overview

In this assignment, you will use the Run I/O console to encrypt and decrypt text using a simple ciphers. Specifically you will encrypt and decrypt text with a Vigenère cipher, which is a simple extension of the well know Caesar cipher, where letters are shifted by a given number. For instance a shift of 3 would change an A to D. At the end of the alphabet, we wrap back to the beginning, so a shift of 3 also changes a Z to C.

We will assume that all text is ASCII, and we will work only with capital letters. The key will also be written using letters where A corresponds to a shift of 0, B to 1, C to 2, and so on. Decryption is done by simply shifting instead by a negative amount.

In a Vigenère cipher, to use a key with multiple letters, we repeat the key over and over again. Where the plain text contains spaces and punctuation, we will leave the text unchanged, and skip to the next letter in the key and the next letter in the text. For instance "LET'S GO TO THE CAT MUSEUM!" with key "AB" will becomes "LFT'S GP UO TIE CBT MVSFUN!", that is, letters in even positions starting at position zero are not shifted, while letters in odd positions are shifted by 1.



There are many interesting ways to try to break Vigenère ciphers, but among the most simple is to look at letter frequencies. In this assignment, you will be required to compute the frequencies of letters to try to identify the key, but this will not be an entirely automatic process. For a given guess, you will need to also specify an assumed length of the key and what letter you believe to be most probable. While the letter E is most common in large blocks of English text, other letters are also very common. See the Figure inset at right.

Provided code and files

You ar provided some code to help you get started, along with a small collection of test files of encrypted text. Once you've completed the objectives of the assignment, you should be able to discover the keys for these encrypted texts. There are no marks associated for discovering the keys, so feel free to discuss the keys and contents of the decrypted text with your classmates!

The provided code implements a simple command menu to let you read a text file into a buffer, print the text in the buffer, encrypt the buffer, decrypt the buffer, write the buffer to a file, guess an encryption key, and quit. The code handles reading from and writing to files and a few other basic procedures to let you focus on the job of writing procedures for encrypting text, decrypting text, and guessing keys. Here follows an example session.

```
Commands (read, print, encrypt, decrypt, write, guess, quit):r
Enter file name:plain0.txt
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.
Commands (read, print, encrypt, decrypt, write, guess, quit):e
Enter key (upper case letters only):B
UIF RVJDL CSPXO GPY KVNQT PWFS UIF MBAZ EPH.
Commands (read, print, encrypt, decrypt, write, guess, quit):w
Enter file name:cipher0.txt
Commands (read, print, encrypt, decrypt, write, guess, quit):g
Enter a nubmer (the key length) for guessing:1
Enter guess of most common letter:O
B
Commands (read, print, encrypt, decrypt, write, guess, quit):p
UIF RVJDL CSPXO GPY KVNQT PWFS UIF MBAZ EPH.
Commands (read, print, encrypt, decrypt, write, guess, quit):d
Enter key (upper case letters only):B
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.
Commands (read, print, encrypt, decrypt, write, guess, quit):q
```

Note that MARS will open files by default in the directory in which it is started. You can place your files in that location for easy reading and writing, or specify the full path, or likewise specify the directory in which to start the MARS jar.

2 EncryptBuffer (3 marks)

Implement the EncryptBuffer procedure. This procedure takes two arguments. The first is a null terminated ASCII string to encrypt (i.e., the text buffer), while the second is a null terminated ASCII string for the key. You can assume that the letters in the buffer are all uppercase, and likewise, that the key is specified with all uppercase letters. You should leave any non-letters unchanged (e.g., punctuation and white spaces).

3 DecryptBuffer (3 marks)

Implement the DecryptBuffer procedure. This procedure takes two arguments. The first is a null terminated ASCII string to decrypt (i.e., the text buffer), while the second is a null terminated ASCII string for the key. You can assume that the letters in the buffer are all uppercase, and likewise, that the key is specified with all uppercase letters. You should leave any non-letters unchanged (e.g., punctuation and white spaces).

4 GuessKey (4 marks)

Implement the GuessKey procedure, and note that you should break down the problem into smaller parts (i.e., write multiple procedures). The procedure takes 4 arguments: the size of the key to guess, the null terminated text buffer, a block of memory in which to write the guessed key (i.e., you are returning the guessed key to the caller in this memory), and finally the last parameter is the letter which you can assume to be most common in the cypher text.

In the example interactive session above, you can see that the guess command was called with a key length of 1 and a common letter guess of O. To guess the key, you must compute letter frequencies. For the string ”UIF RVJDL CSPXO GPY KVNQT PWFS UIF MBAZ EPH”, because the letter P is the most common, this is assumed to map to the letter O, so we guess that the key is B so as to shift all the letters by one position.

You are free to organize your solution as you please, but must use register calling conventions and the stack as necessary. You may find it useful to allocated space in the `.data` segment for an array of integers for counting frequencies. You may find it useful to write a procedure for counting the letter frequencies. More specifically for counting frequencies of letters in the position i where $i \bmod n = k$, that is, for trying to figure out the k^{th} letter of a key where the assumption is that the key is of length n. You may find it useful to write a helper function for zeroing a block of memory (i.e., your count buffer), or likewise identifying a letter within the key given a set of frequencies and a presumed common letter.

Breaking your solution into multiple procedures will help you create an easy to understand solution. Feel free to write additional procedures as necessary or even add commands to the menu that might help (e.g., printing letter frequencies might be useful).