Cryptography (CS362)

Assignment - 1

U19CS012

The below cipher text was generated using **Caesar Cipher**.

```
G JOYGJBGTZGMK UL ZNK MKTKXGR YAHYZOZAZOUT IOVNKX OY ZNGZ HUZN YKTJKX
GTJ XKIKOBKX SAYZ IUSSOZ ZNK VKXSAZKJ IOVNKX YKWAKTIK ZU SKSUXE. G IUSSUT
ZKINTOWAK LUX GBUOJOTM ZNOY OY ZU AYK G QKECUXJ LXUS CNOIN ZNK IOVNKX YKWAKTIK
IGT HK MKTKXGZKJ. LUX KDGSVRK, AYOTM ZNK QKECUXJ IOVNKX, CXOZK UAZ
ZNK QKECUXJ LURRUCKJ HE ATAYKJ RKZZKXY OT TUXSGR UXJKX GTJ SGZIN ZNOY GMGOTYZ
ZNK VRGOTZKDZ RKZZKXY
```

1. Write a program to perform **Brute Force attack** on the given cipher.

Input

Any Cipher-Text generated using Caesar Cipher Encryption algorithm.

Output

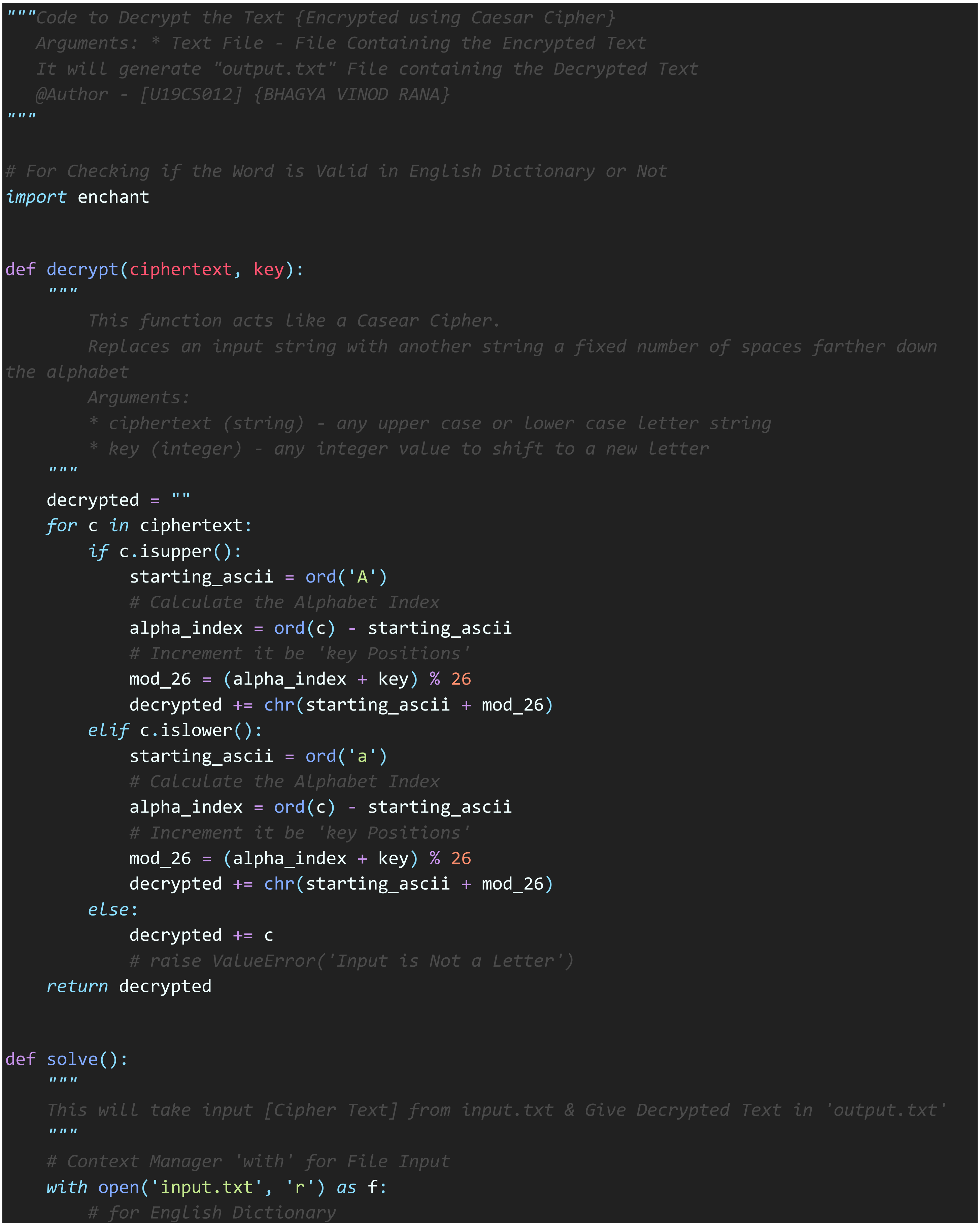
Key value using which cipher text was generated.

Also **Decrypted Message** is Generated in separate File.

Note

- ✓ **Automate** the process of identifying the legitimate plaintext generated from each key e.g. assume that the plaintext was **English text**.
- ✓ Your program should include **Logic** that can identify English text in the **Brute Force attack**.
- ✓ Submit in form of Folder that contain :
- ➢ **Source** code
- ➢ **Executable** file
- ➢ **Steps** to run your program.

Code



```
"""Code to Decrypt the Text {Encrypted using Caesar Cipher}
   Arguments: * Text File - File Containing the Encrypted Text
   It will generate "output.txt" File containing the Decrypted Text
   @Author - [U19CS012] {BHAGYA VINOD RANA}
"""

# For Checking if the Word is Valid in English Dictionary or Not
import enchant


def decrypt(ciphertext, key):
    """
        This function acts like a Casear Cipher.
        Replaces an input string with another string a fixed number of spaces farther down
the alphabet
        Arguments:
        * ciphertext (string) - any upper case or lower case letter string
        * key (integer) - any integer value to shift to a new letter
    """
    decrypted = ""
    for c in ciphertext:
        if c.isupper():
            starting_ascii = ord('A')
            # Calculate the Alphabet Index
            alpha_index = ord(c) - starting_ascii
            # Increment it be 'key Positions'
            mod_26 = (alpha_index + key) % 26
            decrypted += chr(starting_ascii + mod_26)
        elif c.islower():
            starting_ascii = ord('a')
            # Calculate the Alphabet Index
            alpha_index = ord(c) - starting_ascii
            # Increment it be 'key Positions'
            mod_26 = (alpha_index + key) % 26
            decrypted += chr(starting_ascii + mod_26)
        else:
            decrypted += c
            # raise ValueError('Input is Not a Letter')
    return decrypted


def solve():
    """
    This will take input [Cipher Text] from input.txt & Give Decrypted Text in 'output.txt'
    """
    # Context Manager 'with' for File Input
    with open('input.txt', 'r') as f:
        # for English Dictionary
        d = enchant.Dict("en_US")

        # Not used f.read()/f.readlines()/f.readline() - to Avoid Running Out of Memory
        for cryptic_text in f:

            max_valid_token = 0
            final_plain_text = ""
            final_key = 0

            # Brute Force Attack
            for i in range(0, 26):

                plain_text = decrypt(cryptic_text, i)

                # Get Tokens of the Plain Text
                plain_txt_token = plain_text.split()
                # Count the Number of Valid Tokens in Plain Text
                valid_tokens_cnt = 0
                for token in plain_txt_token:
                    if d.check(token) == True:
                        valid_tokens_cnt += 1

                # Update if the valid_token_count is maximum
                if(valid_tokens_cnt > max_valid_token):
                    final_plain_text = plain_text
                    final_key = i
                    max_valid_token = valid_tokens_cnt

            # print("For key {}, Decrypted Text:\n {} \n".format(final_key,
final_plain_text))
            with open("output.txt", "a") as output_file:
                output_file.write(final_plain_text)


# Call to main solve() Function
if __name__ == "__main__":
    solve()
```

Pre-requisites:

- ✓ Python3
- ✓ Enchant Module

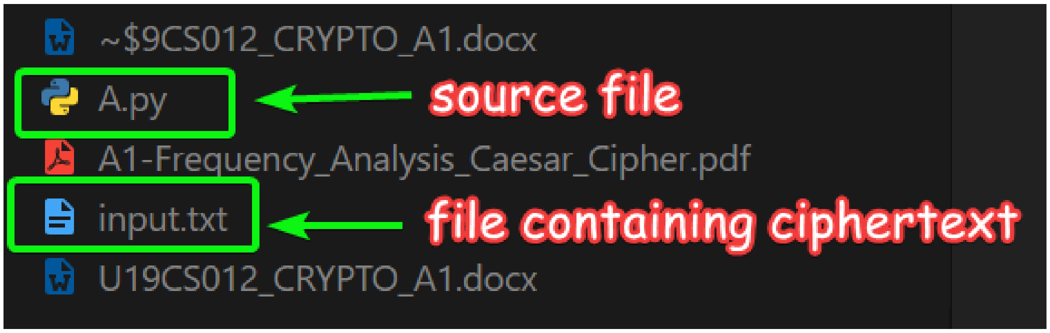
For downloading the **Enchant** Module, use below Command in your Command Line.

```
pip install pyenchant
```

Execution Instruction Steps

1.) Add the "Encrypted" Cipher Text [Caesar Cipher Algorithm] in `input.txt` File

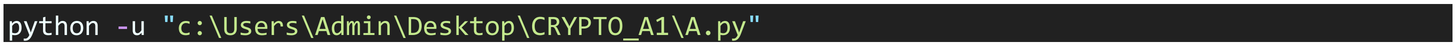
2.) Open Terminal in Folder where Both Source Code {`A.py`} & `input.txt` are present.



source file

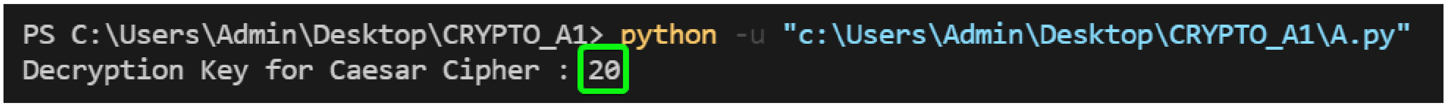
file containing ciphertext

3.) Type the Below Command:



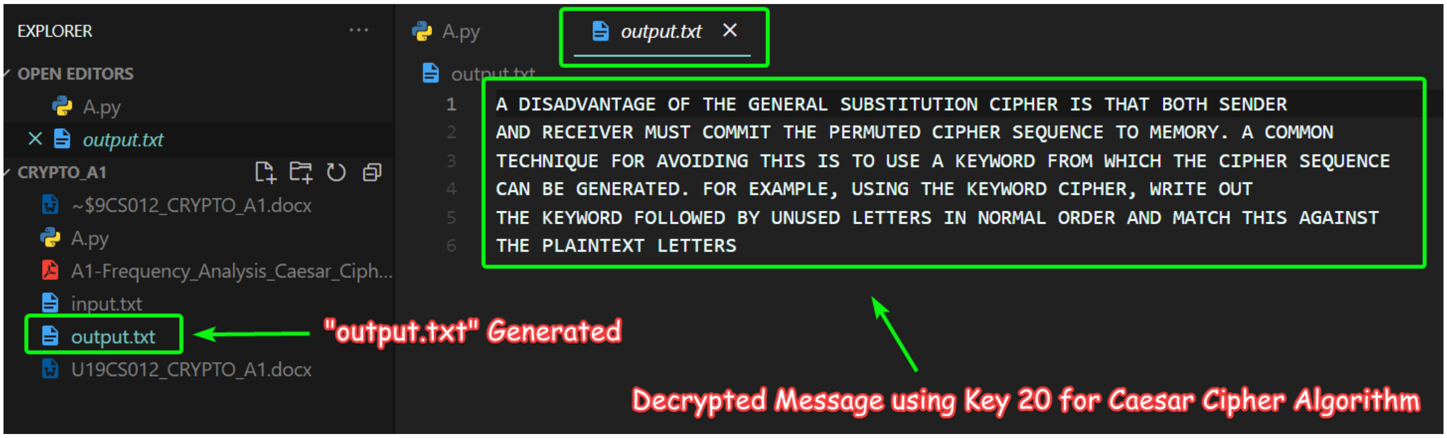
```
python -u "c:\Users\Admin\Desktop\CRYPTO_A1\A.py"
```

As shown below



```
PS C:\Users\Admin\Desktop\CRYPTO_A1> python -u "c:\Users\Admin\Desktop\CRYPTO_A1\A.py"
Decryption Key for Caesar Cipher : 20
```

"output.txt" Generated

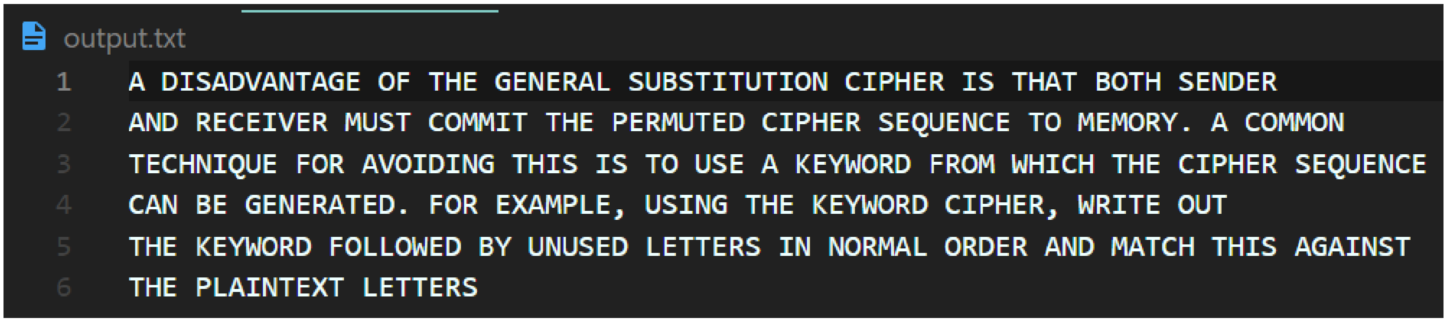


```
A DISADVANTAGE OF THE GENERAL SUBSTITUTION CIPHER IS THAT BOTH SENDER
AND RECEIVER MUST COMMIT THE PERMUTED CIPHER SEQUENCE TO MEMORY. A COMMON
TECHNIQUE FOR AVOIDING THIS IS TO USE A KEYWORD FROM WHICH THE CIPHER SEQUENCE
CAN BE GENERATED. FOR EXAMPLE, USING THE KEYWORD CIPHER, WRITE OUT
THE KEYWORD FOLLOWED BY UNUSED LETTERS IN NORMAL ORDER AND MATCH THIS AGAINST
THE PLAINTEXT LETTERS
```

"output.txt" Generated

Decrypted Message using Key 20 for Caesar Cipher Algorithm

Output



```
A DISADVANTAGE OF THE GENERAL SUBSTITUTION CIPHER IS THAT BOTH SENDER
AND RECEIVER MUST COMMIT THE PERMUTED CIPHER SEQUENCE TO MEMORY. A COMMON
TECHNIQUE FOR AVOIDING THIS IS TO USE A KEYWORD FROM WHICH THE CIPHER SEQUENCE
CAN BE GENERATED. FOR EXAMPLE, USING THE KEYWORD CIPHER, WRITE OUT
THE KEYWORD FOLLOWED BY UNUSED LETTERS IN NORMAL ORDER AND MATCH THIS AGAINST
THE PLAINTEXT LETTERS
```

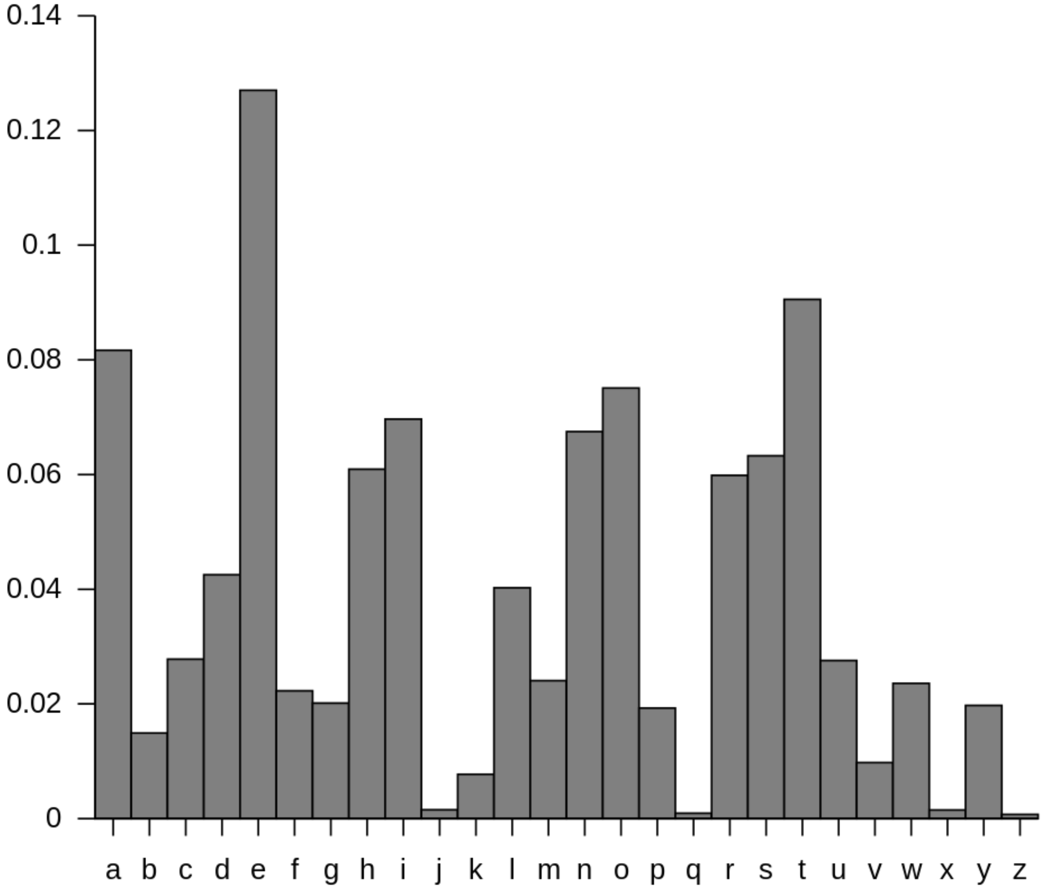
2. To illustrate the use of **Frequency analysis** for breaking the cipher.

You can use the program given in below link which will help you to carry out frequency analysis attack for such cipher text produced by a Mono-Alphabetic cipher. [http://crypto.interactive-maths.com/frequency-analysis-breaking-the-code.html]

Your goal is to find the **Plaintext**, as well as the **key** employed for the above encryption. Clearly explain the **Methodology i.e.** how you could break the code **step by step** while performing the frequency analysis.

Frequencies Analysis: using known character frequencies to **decrypt** a cipher

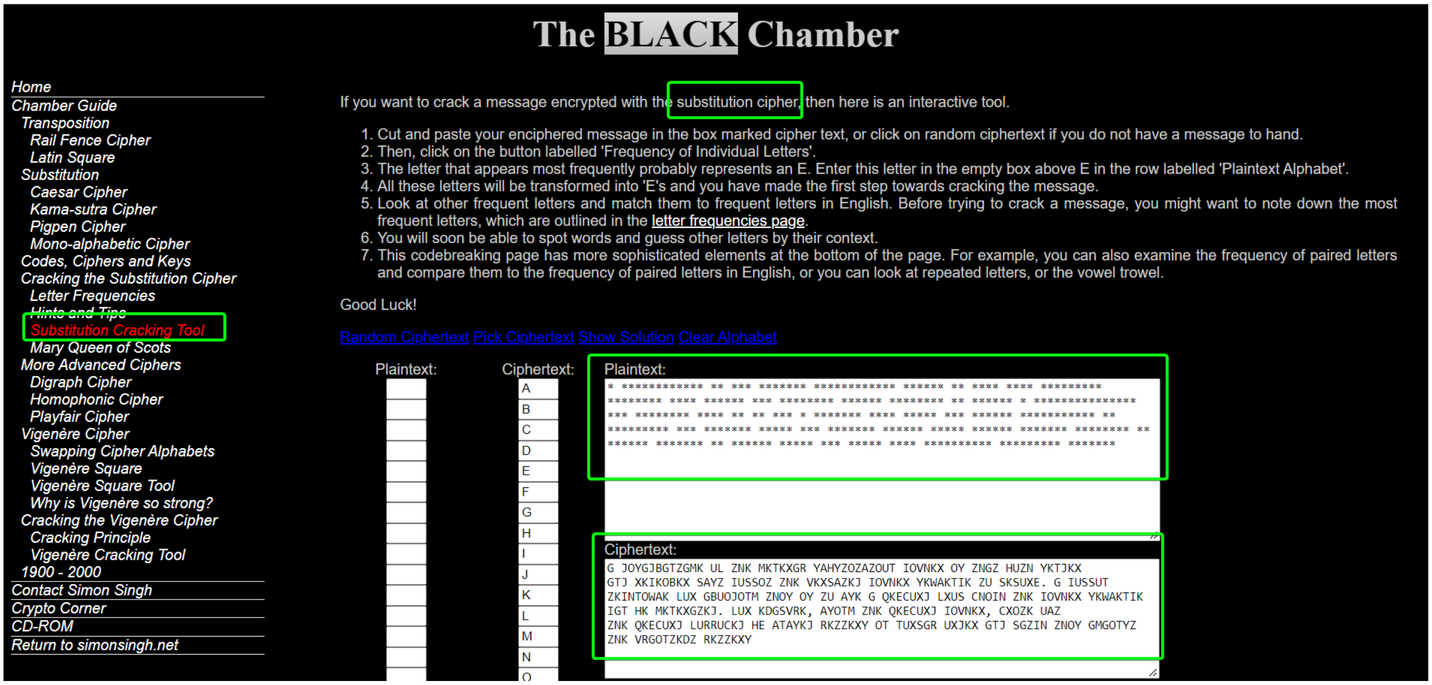
Since, we are working with a Mono-Alphabetic Cipher, we should examine the frequencies of the **letters** of **English** Alphabets.



Above is list of average frequencies for letters in the English language. So, for example, the letter **E** accounts for **12.7%** of all letters in English, whereas **Z** accounts for 0.1 %. {**The Average Distribution**}

1.) We will be using the Below Frequency Analysis Tool {since it does not have Ads}

https://www.simonsingh.net/The_Black_Chamber/substitutioncrackingtool.html



The BLACK Chamber

Home
Chamber Guide
Transposition
Rail Fence Cipher
Latin Square
Substitution
Caesar Cipher
Kama-sutra Cipher
Pigpen Cipher
Mono-alphabetic Cipher
Codes, Ciphers and Keys
Cracking the Substitution Cipher
Letter Frequencies
Hints and Tips
Substitution Cracking Tool
Mary Queen of Scots
More Advanced Ciphers
Digraph Cipher
Homophonic Cipher
Playfair Cipher
Vigenère Cipher
Swapping Cipher Alphabets
Vigenère Square
Vigenère Square Tool
Why is Vigenère so strong?
Cracking the Vigenère Cipher
Cracking Principle
Vigenère Cracking Tool
1900 - 2000
Contact Simon Singh
Crypto Corner
CD-ROM
Return to simonsingh.net

If you want to crack a message encrypted with the substitution cipher, then here is an interactive tool.

1. Cut and paste your enciphered message in the box marked cipher text, or click on random ciphertext if you do not have a message to hand.
2. Then, click on the button labelled 'Frequency of Individual Letters'.
3. The letter that appears most frequently probably represents an E. Enter this letter in the empty box above E in the row labelled 'Plaintext Alphabet'.
4. All these letters will be transformed into 'E's and you have made the first step towards cracking the message.
5. Look at other frequent letters and match them to frequent letters in English. Before trying to crack a message, you might want to note down the most frequent letters, which are outlined in the letter frequencies page.
6. You will soon be able to spot words and guess other letters by their context.
7. This codebreaking page has more sophisticated elements at the bottom of the page. For example, you can also examine the frequency of paired letters and compare them to the frequency of paired letters in English, or you can look at repeated letters, or the vowel trowel.

Good Luck!

Random Ciphertext Pick Ciphertext Show Solution Clear Alphabet

Plaintext: Ciphertext:
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O

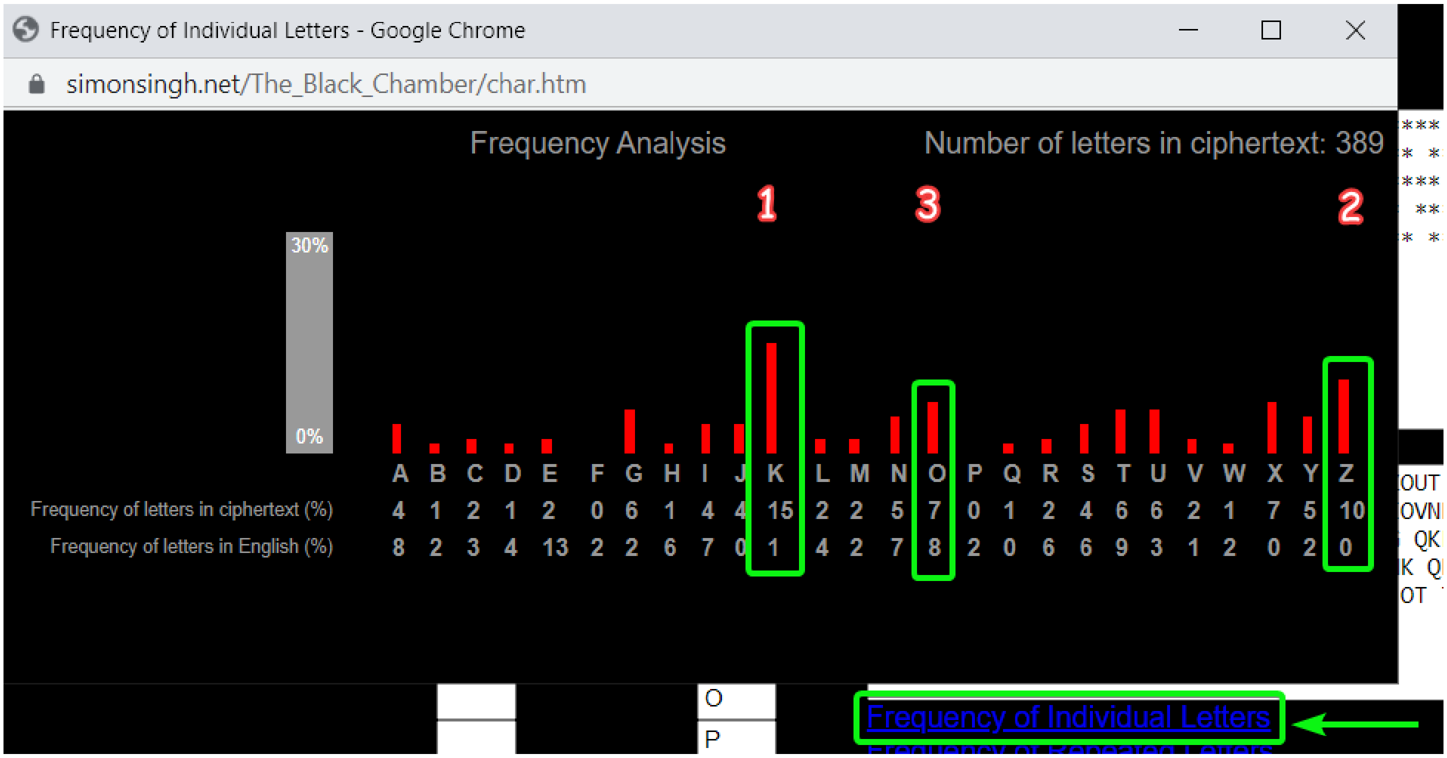
Plaintext:

```
* ************ ** *** ******* ************ ****** ** **** **** *********
******** **** ****** *** ******** ****** ******** ** ****** * ***************
*** ******** **** ** ** *** * ******* **** ***** *** ****** *********** **
********* *** ******* ***** *** ******* ****** ***** ****** ******* ******** **
****** ******* ** ****** ***** *** ***** **** ********** ********* *******
```

Ciphertext:

```
G JOYGJBGTZGMK UL ZNK MKTKXGR YAHYZOZAZOUT IOVNKX OY ZNGZ HUZN YKTJKX
GTJ XKIKOBKX SAYZ IUSSOZ ZNK VKXSAZKJ IOVNKX YKWAKTIK ZU SKSUXE. G IUSSUT
ZKINTOWAK LUX GBUOJOTM ZNOY OY ZU AYK G QKECUXJ LXUS CNOIN ZNK IOVNKX YKWAKTIK
IGT HK MKTKXGZKJ. LUX KDGSVRK, AYOTM ZNK QKECUXJ IOVNKX, CXOZK UAZ
ZNK QKECUXJ LURRUCKJ HE ATAYKJ RKZZKXY OT TUXSGR UXJKX GTJ SGZIN ZNOY GMGOTYZ
ZNK VRGOTZKDZ RKZZKXY
```

2.) Watch the Frequency of Individual Letters



Frequency of Individual Letters - Google Chrome

simonsingh.net/The_Black_Chamber/char.htm

Frequency Analysis

Number of letters in ciphertext: 389

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Frequency of letters in ciphertext (%)	4	1	2	1	2	0	6	1	4	4	15	2	2	5	7	0	1	2	4	6	6	2	1	7	5	10
Frequency of letters in English (%)	8	2	3	4	13	2	2	6	7	0	1	4	2	7	8	2	0	6	6	9	3	1	2	0	2	0

Frequency of Individual Letters

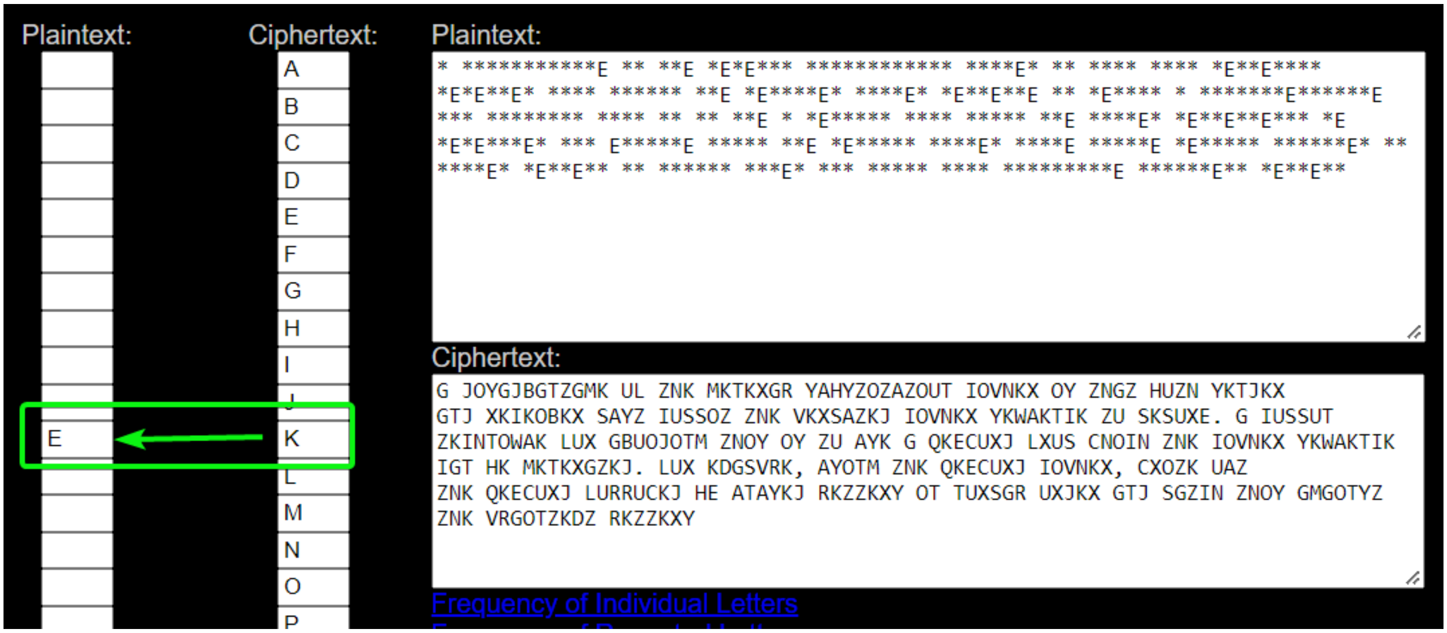
Frequency of Repeated Letters

We can observe that Letter "K" has maximum frequency [15], followed by Letter "Z" [10], and then Letter "O".

Observations:

(a) Since Letter "K" is most Frequent in Given Cipher Text & If we consider Average Distribution of English Alphabet Frequencies, Letter "E" is most Frequent. {It may/may not be Right Substitution}

∴ K -> E



Plaintext:	Ciphertext:
	A
	B
	C
	D
	E
	F
	G
	H
	I
	J
E	K
	L
	M
	N
	O
	P

Plaintext:

```
* ***********E ** **E *E*E*** ************ ****E* ** **** **** *E**E****
*E*E**E* **** ****** **E *E****E* ****E* *E**E**E ** *E**** * *******E******E
*** ******** **** ** ** **E * *E***** **** ***** **E ****E* *E**E**E*** *E
*E*E***E* *** E*****E ***** **E *E***** ****E* ****E *****E *E***** ******E* **
****E* *E**E** ** ****** ***E* *** ***** **** *********E ******E** *E**E**
```

Ciphertext:

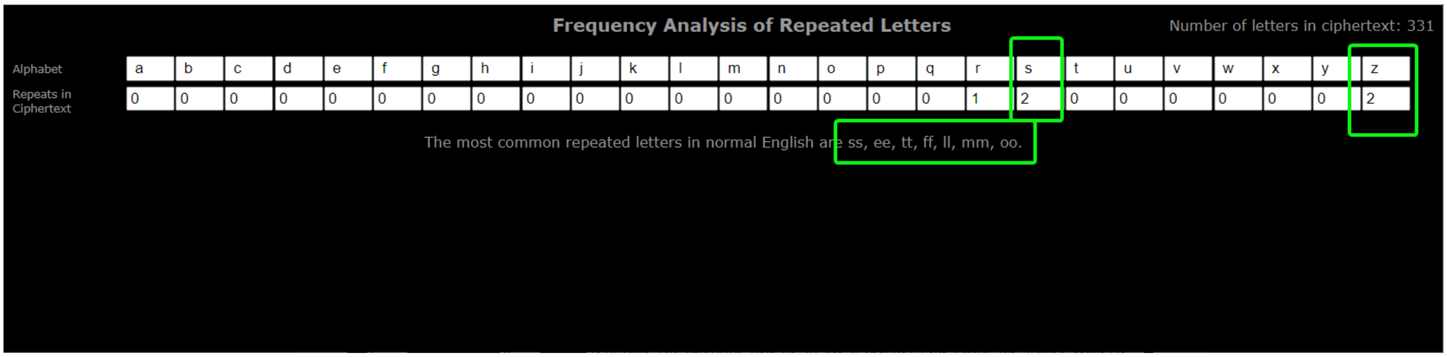
```
G JOYGJBGTZGMK UL ZNK MKTKXGR YAHYZOZAZOUT IOVNKX OY ZNGZ HUZN YKTJKX
GTJ XKIKOBKX SAYZ IUSSOZ ZNK VKXSAZKJ IOVNKX YKWAKTIK ZU SKSUXE. G IUSSUT
ZKINTOWAK LUX GBUOJOTM ZNOY OY ZU AYK G QKECUXJ LXUS CNOIN ZNK IOVNKX YKWAKTIK
IGT HK MKTKXGZKJ. LUX KDGSVRK, AYOTM ZNK QKECUXJ IOVNKX, CXOZK UAZ
ZNK QKECUXJ LURRUCKJ HE ATAYKJ RKZZKXY OT TUXSGR UXJKX GTJ SGZIN ZNOY GMGOTYZ
ZNK VRGOTZKDZ RKZZKXY
```

Frequency of Individual Letters

We don't see any Abnormal English Words being generated by this Substitution.

(b) Since Letter "Z" is **Next** most Frequent in Given Cipher Text & If we consider Average Distribution of English Alphabet Frequencies, Letter "T" is Next most Frequent. {It may/may not be Right Substitution}

∴ Z -> T

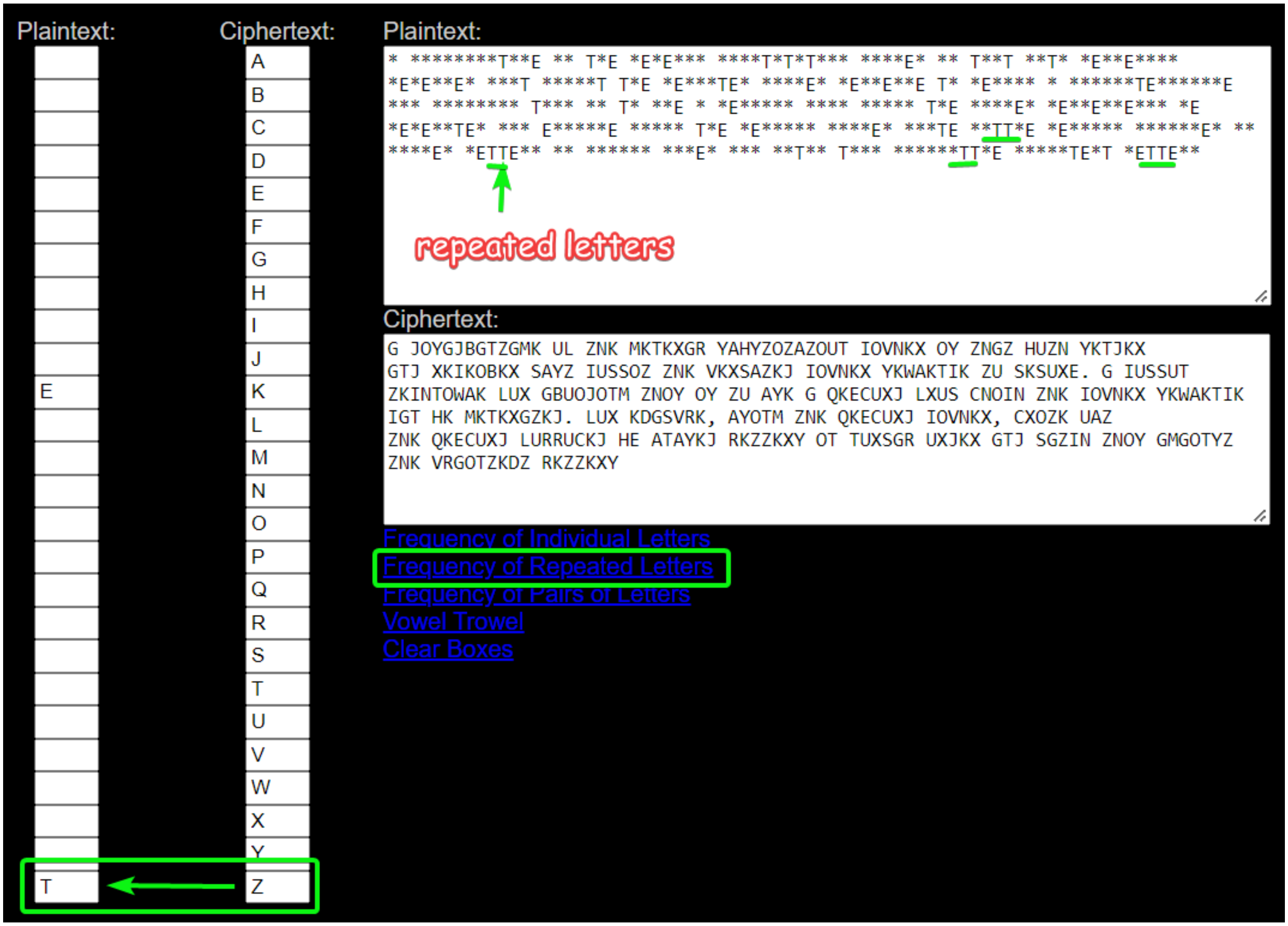


Frequency Analysis of Repeated Letters

Number of letters in ciphertext: 331

Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Repeats in Ciphertext	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	2

The most common repeated letters in normal English are ss, ee, tt, ff, ll, mm, oo.



Plaintext:	Ciphertext:
	A
	B
	C
	D
	E
	F
	G
	H
	I
	J
E	K
	L
	M
	N
	O
	P
	Q
	R
	S
	T
	U
	V
	W
	X
	Y
T	Z

Plaintext:

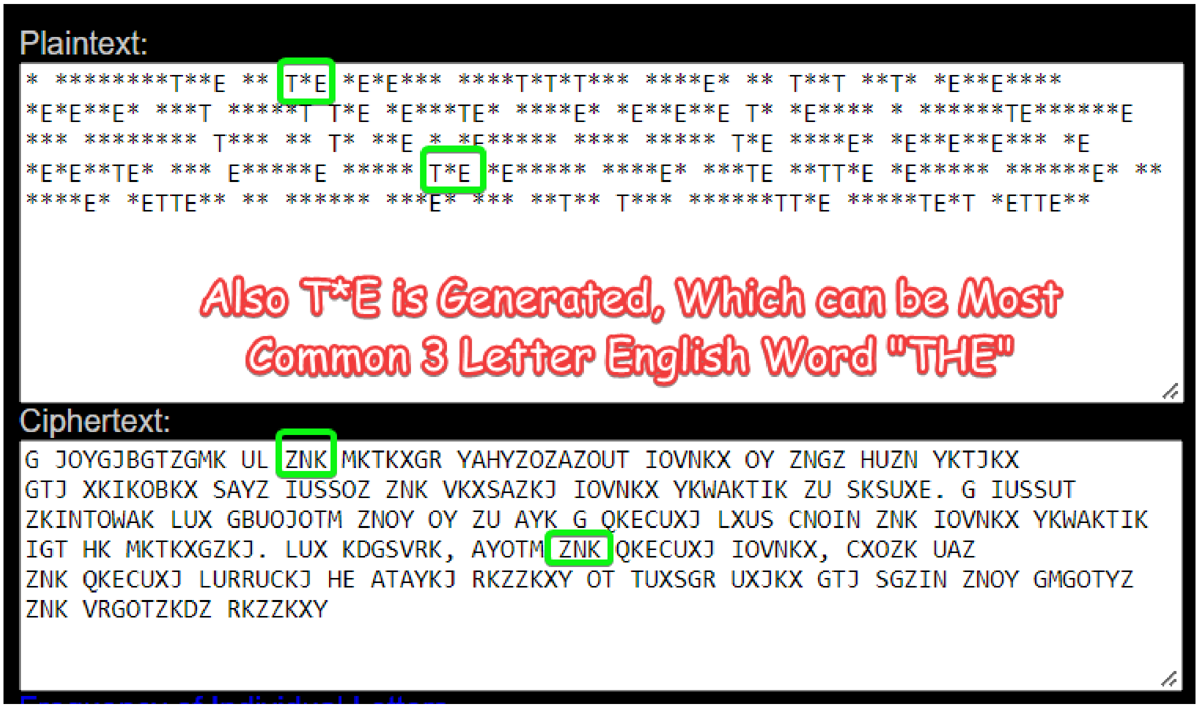
```
* ********T**E ** T*E *E*E*** ****T*T*T*** ****E* ** T**T **T* *E**E****
*E*E**E* ***T *****T T*E *E***TE* ****E* *E**E**E T* *E**** * ******TE******E
*** ******** T*** ** T* **E * *E***** **** ***** T*E ****E* *E**E**E*** *E
*E*E**TE* *** E*****E ***** T*E *E***** ****E* ***TE **TT*E *E***** ******E* **
****E* *ETTE** ** ****** ***E* *** **T** T*** *****TT*E *****TE*T *ETTE**
```

repeated letters

Ciphertext:

```
G JOYGJBGTZGMK UL ZNK MKTKXGR YAHYZOZAZOUT IOVNKX OY ZNGZ HUZN YKTJKX
GTJ XKIKOBKX SAYZ IUSSOZ ZNK VKXSAZKJ IOVNKX YKWAKTIK ZU SKSUXE. G IUSSUT
ZKINTOWAK LUX GBUOJOTM ZNOY OY ZU AYK G QKECUXJ LXUS CNOIN ZNK IOVNKX YKWAKTIK
IGT HK MKTKXGZKJ. LUX KDGSVRK, AYOTM ZNK QKECUXJ IOVNKX, CXOZK UAZ
ZNK QKECUXJ LURRUCKJ HE ATAYKJ RKZZKXY OT TUXSGR UXJKX GTJ SGZIN ZNOY GMGOTYZ
ZNK VRGOTZKDZ RKZZKXY
```

Frequency of Individual Letters
Frequency of Repeated Letters
Frequency of Pairs of Letters
Vowel Trowel
Clear Boxes



Plaintext:

```
* ********T**E ** T*E *E*E*** ****T*T*T*** ****E* ** T**T **T* *E**E****
*E*E**E* ***T *****T T*E *E***TE* ****E* *E**E**E T* *E**** * ******TE******E
*** ******** T*** ** T* **E * *E***** **** ***** T*E ****E* *E**E**E*** *E
*E*E**TE* *** E*****E ***** T*E *E***** ****E* ***TE **TT*E *E***** ******E* **
****E* *ETTE** ** ****** ***E* *** **T** T*** *****TT*E *****TE*T *ETTE**
```

Also T*E is Generated, Which can be Most Common 3 Letter English Word "THE"

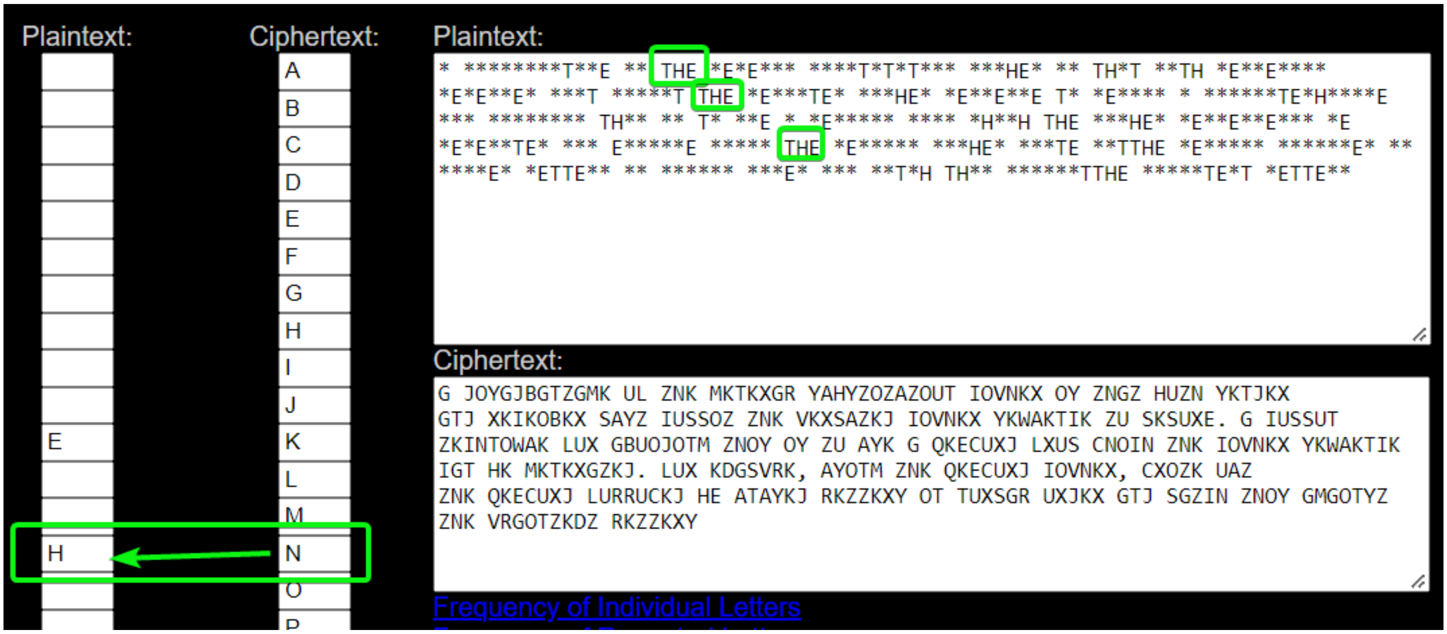
Ciphertext:

```
G JOYGJBGTZGMK UL ZNK MKTKXGR YAHYZOZAZOUT IOVNKX OY ZNGZ HUZN YKTJKX
GTJ XKIKOBKX SAYZ IUSSOZ ZNK VKXSAZKJ IOVNKX YKWAKTIK ZU SKSUXE. G IUSSUT
ZKINTOWAK LUX GBUOJOTM ZNOY OY ZU AYK G QKECUXJ LXUS CNOIN ZNK IOVNKX YKWAKTIK
IGT HK MKTKXGZKJ. LUX KDGSVRK, AYOTM ZNK QKECUXJ IOVNKX, CXOZK UAZ
ZNK QKECUXJ LURRUCKJ HE ATAYKJ RKZZKXY OT TUXSGR UXJKX GTJ SGZIN ZNOY GMGOTYZ
ZNK VRGOTZKDZ RKZZKXY
```

Since, Z -> T, also leads to Most Common Repeated Letters in Normal English & 3 Letter Frequent Word Occurrence, This Substitution is also Fine.

(c) From Observation (b), T*E [Plaintext] -> ZNK [Cipher],

∴ N -> H



Plaintext:	Ciphertext:
	A
	B
	C
	D
	E
	F
	G
	H
	I
	J
E	K
	L
	M
H	N
	O
	P

Plaintext:

```
* ********T**E ** THE *E*E*** ****T*T*T*** ***HE* ** TH*T **TH *E**E****
*E*E**E* ***T *****T THE *E***TE* ***HE* *E**E**E T* *E**** * ******TE*H****E
*** ******** TH** ** T* **E * *E***** **** *H**H THE ***HE* *E**E**E*** *E
*E*E**TE* *** E*****E ***** THE *E***** ***HE* ***TE **TTHE *E***** ******E* **
****E* *ETTE** ** ****** ***E* *** **T*H TH** ******TTHE *****TE*T *ETTE**
```

Ciphertext:

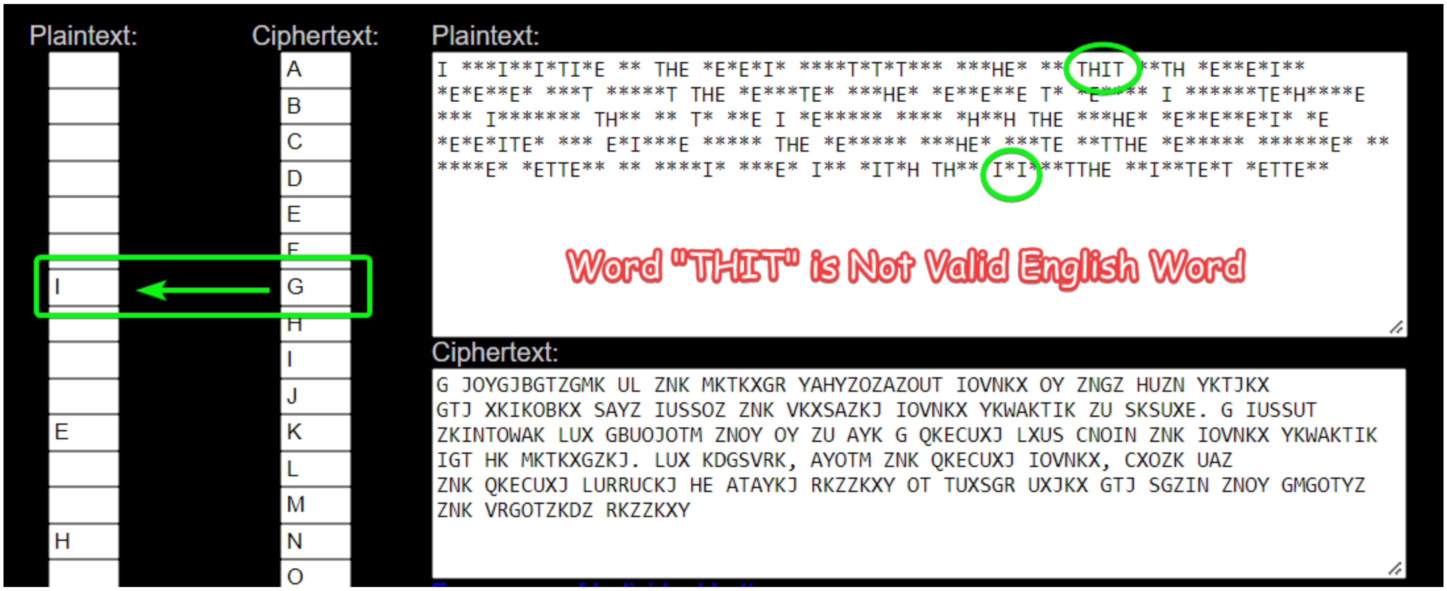
```
G JOYGJBGTZGMK UL ZNK MKTKXGR YAHYZOZAZOUT IOVNKX OY ZNGZ HUZN YKTJKX
GTJ XKIKOBKX SAYZ IUSSOZ ZNK VKXSAZKJ IOVNKX YKWAKTIK ZU SKSUXE. G IUSSUT
ZKINTOWAK LUX GBUOJOTM ZNOY OY ZU AYK G QKECUXJ LXUS CNOIN ZNK IOVNKX YKWAKTIK
IGT HK MKTKXGZKJ. LUX KDGSVRK, AYOTM ZNK QKECUXJ IOVNKX, CXOZK UAZ
ZNK QKECUXJ LURRUCKJ HE ATAYKJ RKZZKXY OT TUXSGR UXJKX GTJ SGZIN ZNOY GMGOTYZ
ZNK VRGOTZKDZ RKZZKXY
```

Frequency of Individual Letters

(d) Since One-letter words in English are "A" and "I",

∴ We can Safely Predict that Letter "G" -> Either "A" or "I".

Lets Try G->I



Plaintext:	Ciphertext:
	A
	B
	C
	D
	E
	F
I	G
	H
	I
	J
E	K
	L
	M
H	N
	O

Plaintext:

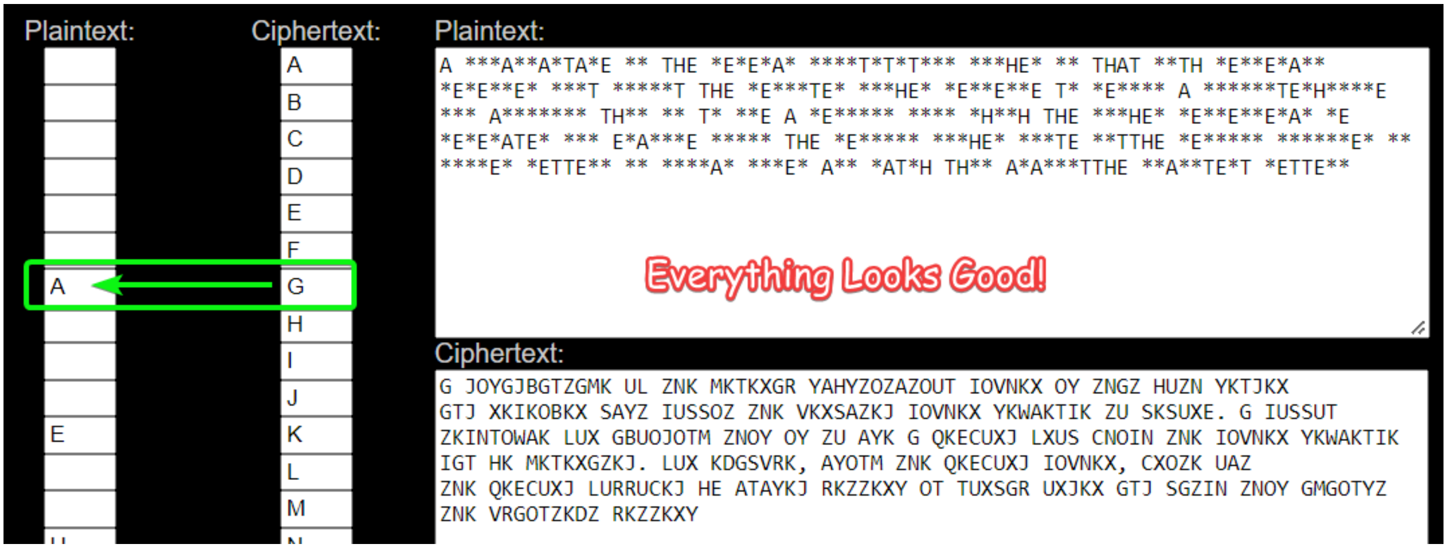
```
I ***I**I*TI*E ** THE *E*E*I* ****T*T*T*** ***HE* ** THIT **TH *E**E*I**
*E*E**E* ***T *****T THE *E***TE* ***HE* *E**E**E T* *E**** I ******TE*H****E
*** I******* TH** ** T* **E I *E***** **** *H**H THE ***HE* *E**E**E*I* *E
*E*E*ITE* *** E*I***E ***** THE *E***** ***HE* ***TE **TTHE *E***** ******E* **
****E* *ETTE** ** ****I* ***E* I** *IT*H TH** I*I***TTHE **I**TE*T *ETTE**
```

Word "THIT" is Not Valid English Word

Ciphertext:

```
G JOYGJBGTZGMK UL ZNK MKTKXGR YAHYZOZAZOUT IOVNKX OY ZNGZ HUZN YKTJKX
GTJ XKIKOBKX SAYZ IUSSOZ ZNK VKXSAZKJ IOVNKX YKWAKTIK ZU SKSUXE. G IUSSUT
ZKINTOWAK LUX GBUOJOTM ZNOY OY ZU AYK G QKECUXJ LXUS CNOIN ZNK IOVNKX YKWAKTIK
IGT HK MKTKXGZKJ. LUX KDGSVRK, AYOTM ZNK QKECUXJ IOVNKX, CXOZK UAZ
ZNK QKECUXJ LURRUCKJ HE ATAYKJ RKZZKXY OT TUXSGR UXJKX GTJ SGZIN ZNOY GMGOTYZ
ZNK VRGOTZKDZ RKZZKXY
```

Since, Word "THIT" is Not Valid English Word, ∴ Substituting G->A would Lead to "THAT" which is Valid English Word. ∴ G -> A



Plaintext:	Ciphertext:
	A
	B
	C
	D
	E
	F
A	G
	H
	I
	J
E	K
	L
	M
H	N

Plaintext:

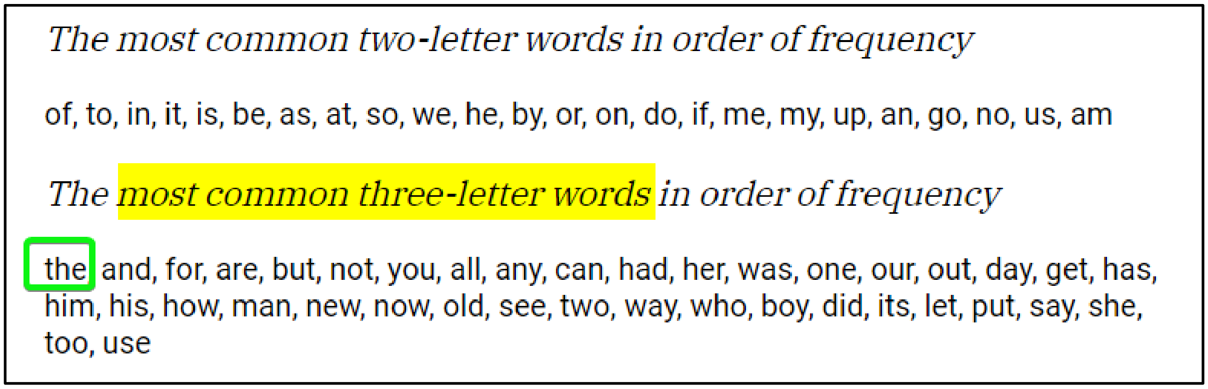
```
A ***A**A*TA*E ** THE *E*E*A* ****T*T*T*** ***HE* ** THAT **TH *E**E*A**
*E*E**E* ***T *****T THE *E***TE* ***HE* *E**E**E T* *E**** A ******TE*H****E
*** A******* TH** ** T* **E A *E***** **** *H**H THE ***HE* *E**E**E*A* *E
*E*E*ATE* *** E*A***E ***** THE *E***** ***HE* ***TE **TTHE *E***** ******E* **
****E* *ETTE** ** ****A* ***E* A** *AT*H TH** A*A***TTHE **A**TE*T *ETTE**
```

Everything Looks Good!

Ciphertext:

```
G JOYGJBGTZGMK UL ZNK MKTKXGR YAHYZOZAZOUT IOVNKX OY ZNGZ HUZN YKTJKX
GTJ XKIKOBKX SAYZ IUSSOZ ZNK VKXSAZKJ IOVNKX YKWAKTIK ZU SKSUXE. G IUSSUT
ZKINTOWAK LUX GBUOJOTM ZNOY OY ZU AYK G QKECUXJ LXUS CNOIN ZNK IOVNKX YKWAKTIK
IGT HK MKTKXGZKJ. LUX KDGSVRK, AYOTM ZNK QKECUXJ IOVNKX, CXOZK UAZ
ZNK QKECUXJ LURRUCKJ HE ATAYKJ RKZZKXY OT TUXSGR UXJKX GTJ SGZIN ZNOY GMGOTYZ
ZNK VRGOTZKDZ RKZZKXY
```

[Most Frequent Substitution - https://scottbryce.com/cryptograms/stats.html]



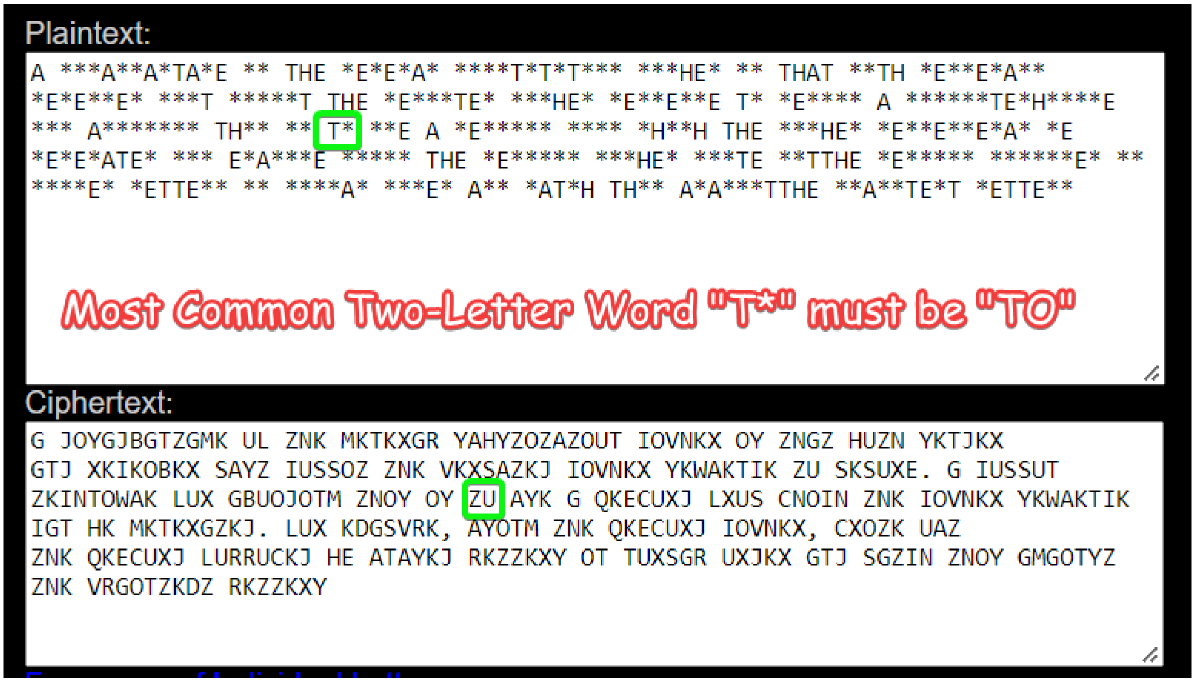
The most common two-letter words in order of frequency

of, to, in, it, is, be, as, at, so, we, he, by, or, on, do, if, me, my, up, an, go, no, us, am

The most common three-letter words in order of frequency

the and, for, are, but, not, you, all, any, can, had, her, was, one, our, out, day, get, has, him, his, how, man, new, now, old, see, two, way, who, boy, did, its, let, put, say, she, too, use

(e) Most Common Two Letter Word Starting with "T" is "TO"



Plaintext:

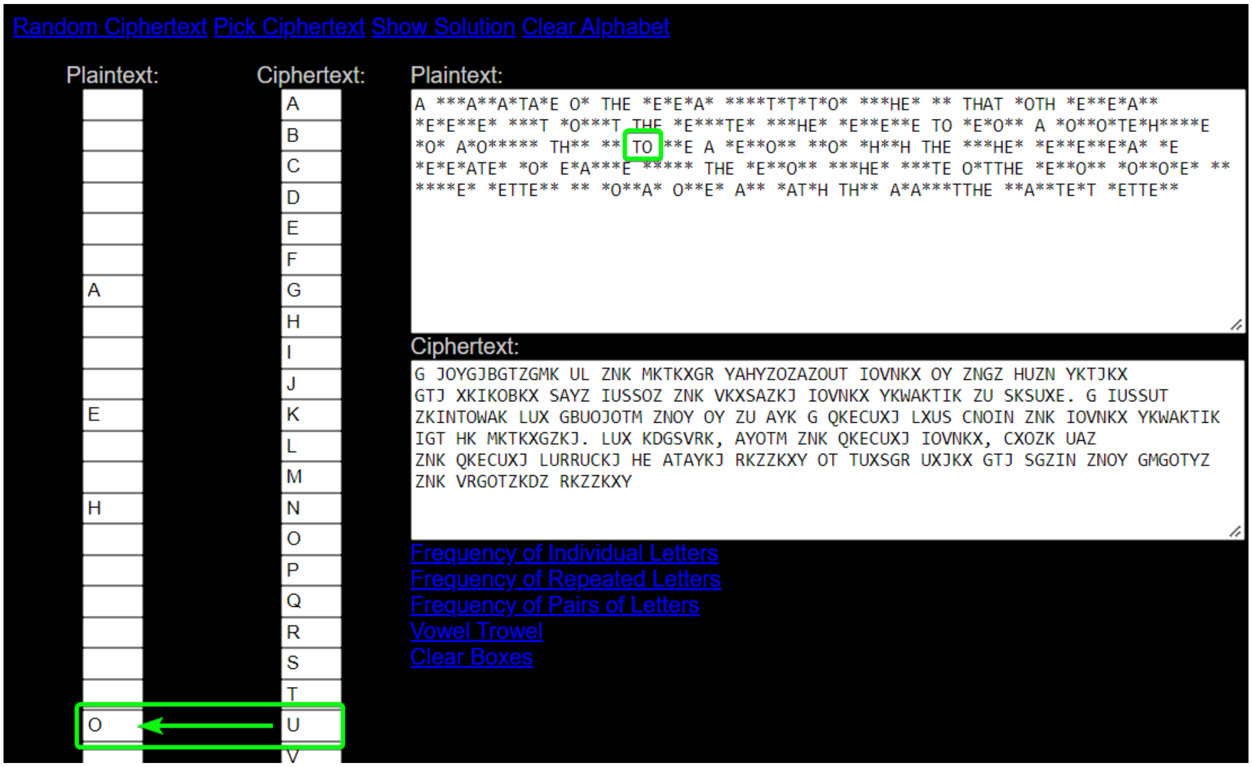
```
A ***A**A*TA*E ** THE *E*E*A* ****T*T*T*** ***HE* ** THAT **TH *E**E*A**
*E*E**E* ***T *****T THE *E***TE* ***HE* *E**E**E T* *E**** A ******TE*H****E
*** A******* TH** ** T* **E A *E***** **** *H**H THE ***HE* *E**E**E*A* *E
*E*E*ATE* *** E*A***E ***** THE *E***** ***HE* ***TE **TTHE *E***** ******E* **
****E* *ETTE** ** ****A* ***E* A** *AT*H TH** A*A***TTHE **A**TE*T *ETTE**
```

Most Common Two-Letter Word "T*" must be "TO"

Ciphertext:

```
G JOYGJBGTZGMK UL ZNK MKTKXGR YAHYZOZAZOUT IOVNKX OY ZNGZ HUZN YKTJKX
GTJ XKIKOBKX SAYZ IUSSOZ ZNK VKXSAZKJ IOVNKX YKWAKTIK ZU SKSUXE. G IUSSUT
ZKINTOWAK LUX GBUOJOTM ZNOY OY ZU AYK G QKECUXJ LXUS CNOIN ZNK IOVNKX YKWAKTIK
IGT HK MKTKXGZKJ. LUX KDGSVRK, AYOTM ZNK QKECUXJ IOVNKX, CXOZK UAZ
ZNK QKECUXJ LURRUCKJ HE ATAYKJ RKZZKXY OT TUXSGR UXJKX GTJ SGZIN ZNOY GMGOTYZ
ZNK VRGOTZKDZ RKZZKXY
```

∴ "TO" -> "ZU", U -> O is Valid Substitution.


Random Ciphertext
Pick Ciphertext
Show Solution
Clear Alphabet
Frequency of Individual Letters
Frequency of Repeated Letters
Frequency of Pairs of Letters
Vowel Trowel
Clear Boxes

Plaintext:	Ciphertext:
	A
	B
	C
	D
	E
	F
A	G
	H
	I
	J
E	K
	L
	M
H	N
	O
	P
	Q
	R
	S
	T
O	U
	V

Plaintext:

```
A ***A**A*TA*E O* THE *E*E*A* ****T*T*T*O* ***HE* ** THAT *OTH *E**E*A**
*E*E**E* ***T *O***T THE *E***TE* ***HE* *E**E**E TO *E*O** A *O**O*TE*H****E
*O* A*O***** TH** ** TO **E A *E**O** **O* *H**H THE ***HE* *E**E**E*A* *E
*E*E*ATE* *O* E*A***E ***** THE *E**O** ***HE* ***TE O*TTHE *E**O** *O**O*E* **
****E* *ETTE** ** *O**A* O**E* A** *AT*H TH** A*A***TTHE **A**TE*T *ETTE**
```

Ciphertext:

```
G JOYGJBGTZGMK UL ZNK MKTKXGR YAHYZOZAZOUT IOVNKX OY ZNGZ HUZN YKTJKX
GTJ XKIKOBKX SAYZ IUSSOZ ZNK VKXSAZKJ IOVNKX YKWAKTIK ZU SKSUXE. G IUSSUT
ZKINTOWAK LUX GBUOJOTM ZNOY OY ZU AYK G QKECUXJ LXUS CNOIN ZNK IOVNKX YKWAKTIK
IGT HK MKTKXGZKJ. LUX KDGSVRK, AYOTM ZNK QKECUXJ IOVNKX, CXOZK UAZ
ZNK QKECUXJ LURRUCKJ HE ATAYKJ RKZZKXY OT TUXSGR UXJKX GTJ SGZIN ZNOY GMGOTYZ
ZNK VRGOTZKDZ RKZZKXY
```

Similarity, "*E" -> HK, & Most Common Two Letter Word Ending with E is "BE".

The most common two-letter words in order of frequency

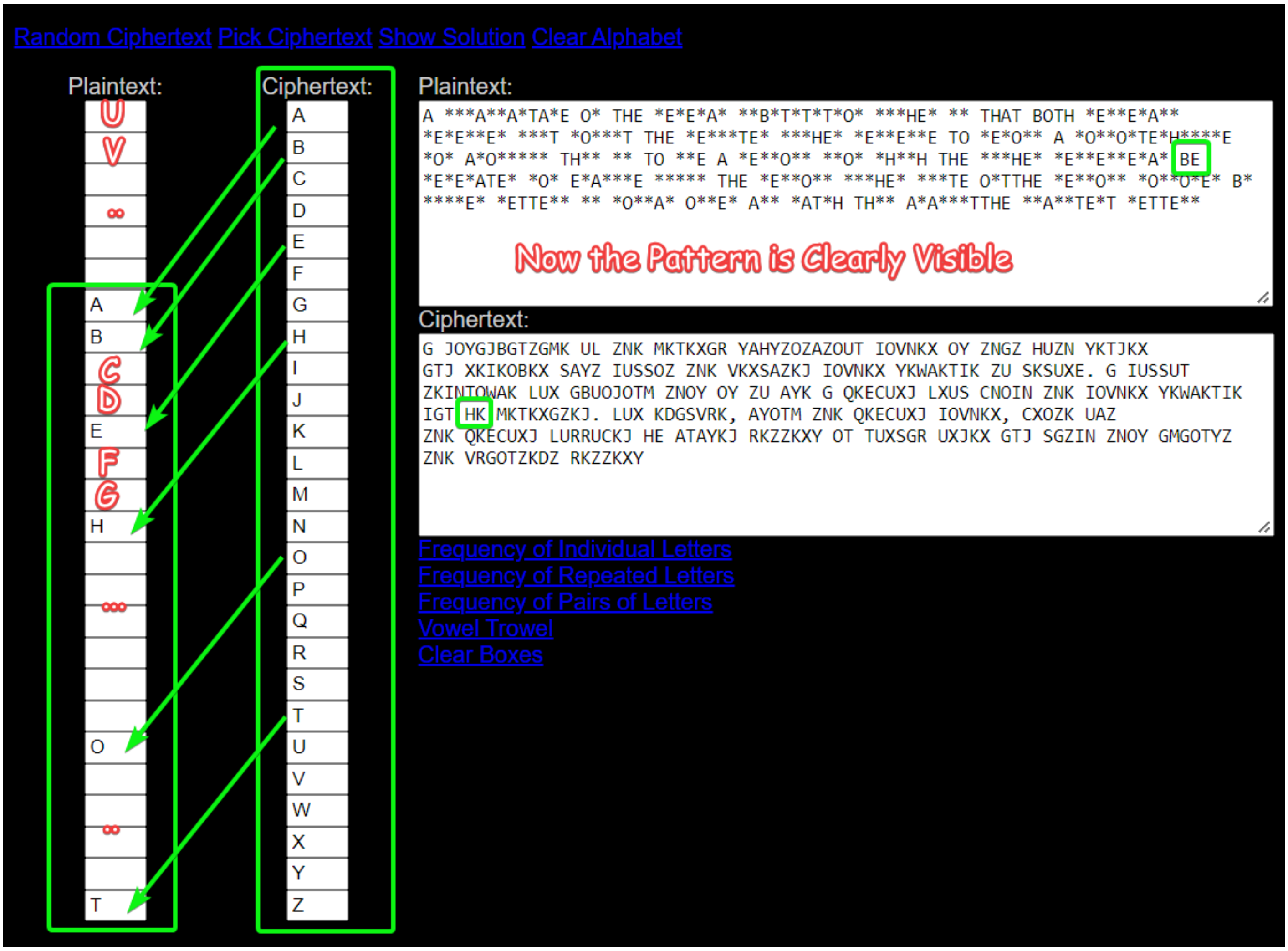
of, to, in, it, is be as, at, so, we, he, by, or, on, do, if, me, my, up, an, go, no, us, am

∴ "BE" -> "HK", H -> B is Valid Substitution.

Cipher-Letter	Plain-Text	Key {Shift} (c + key) % 26
K	E	20
Z	T	20
N	H	20
G	A	20
U	O	20
H	B	20

Therefore, the Pattern is **Clearly Visible** since its Mono-Alphabetic Caeser.

{We got the Key in First Observation itself, but other observations made first Claim Strong.}



Plaintext:

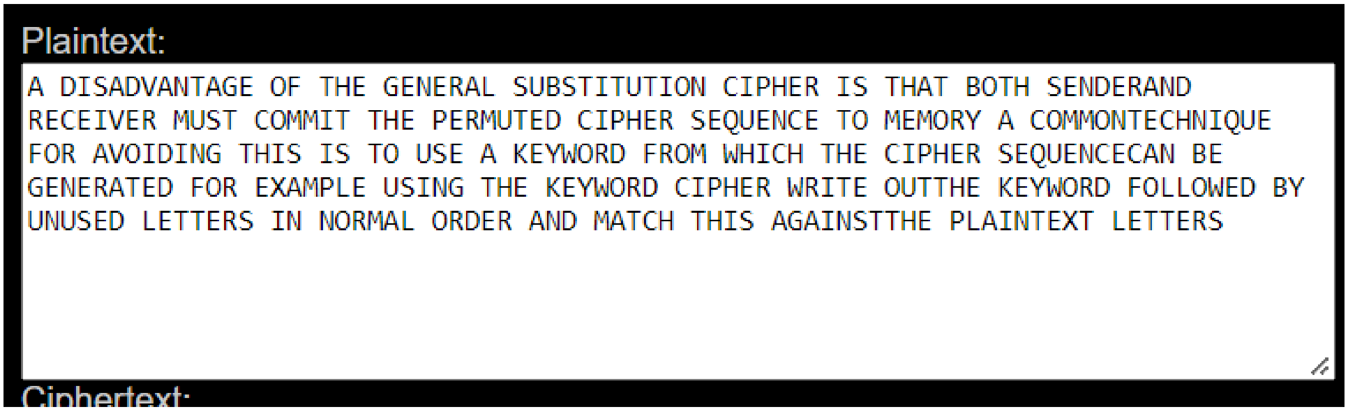
```
A ***A**A*TA*E O* THE *E*E*A* **B*T*T*T*O* ***HE* ** THAT BOTH *E**E*A**
*E*E**E* ***T *O***T THE *E***TE* ***HE* *E**E**E TO *E*O** A *O**O*TE*H****E
*O* A*O***** TH** ** TO **E A *E**O** **O* *H**H THE ***HE* *E**E**E*A* BE
*E*E*ATE* *O* E*A***E ***** THE *E**O** ***HE* ***TE O*TTHE *E**O** *O**O*E* B*
****E* *ETTE** ** *O**A* O**E* A** *AT*H TH** A*A***TTHE **A**TE*T *ETTE**
```

Now the Pattern is Clearly Visible

Ciphertext:

```
G JOYGJBGTZGMK UL ZNK MKTKXGR YAHYZOZAZOUT IOVNKX OY ZNGZ HUZN YKTJKX
GTJ XKIKOBKX SAYZ IUSSOZ ZNK VKXSAZKJ IOVNKX YKWAKTIK ZU SKSUXE. G IUSSUT
ZKINTOWAK LUX GBUOJOTM ZNOY OY ZU AYK G QKECUXJ LXUS CNOIN ZNK IOVNKX YKWAKTIK
IGT HK MKTKXGZKJ. LUX KDGSVRK, AYOTM ZNK QKECUXJ IOVNKX, CXOZK UAZ
ZNK QKECUXJ LURRUCKJ HE ATAYKJ RKZZKXY OT TUXSGR UXJKX GTJ SGZIN ZNOY GMGOTYZ
ZNK VRGOTZKDZ RKZZKXY
```

Message



Plaintext:

```
A DISADVANTAGE OF THE GENERAL SUBSTITUTION CIPHER IS THAT BOTH SENDERAND
RECEIVER MUST COMMIT THE PERMUTED CIPHER SEQUENCE TO MEMORY A COMMONTECHNIQUE
FOR AVOIDING THIS IS TO USE A KEYWORD FROM WHICH THE CIPHER SEQUENCECAN BE
GENERATED FOR EXAMPLE USING THE KEYWORD CIPHER WRITE OUTTHE KEYWORD FOLLOWED BY
UNUSED LETTERS IN NORMAL ORDER AND MATCH THIS AGAINSTTHE PLAINTEXT LETTERS
```

Ciphertext:

After all Substitutions are made, the Cipher Text is **Successfully Decrypted**!

SUBMITTED BY: U19CS012

BHAGYA VINOD RANA