Q1 Commands
5 Points

List the commands used in the game to reach the first ciphertext.

```
list
go
read
enter
read
```

Q2 Cryptosystem
5 Points

What cryptosystem was used at this level?

The CryptoSystem used in this level is Substitution Cipher .The Substitution cipher used in this level is Monoalphabetic Substitution Cipher because here we are replacing each letter/digits/punctuation marks with only one character.

In Monoalphabetic substitution cipher what happens is a letter is always substituted by a particular letter only.
For example: Suppose ‘a’ can be enciphered as ‘d’ everywhere then this is monoalphabetic cipher.

 In Polyalphabetic substitution cipher what happens is a letter is substituted by different different letter.
For example: For example, ‘a’ can be enciphered as ‘d’ in the starting of the text, but as ‘n’ at the middle.

In this Level we have Substitution Cipher(Mono-Alphabetic)

Q3 Analysis
25 Points

What tools and observations were used to figure out the cryptosystem?
NOTE: Failing to provide proper analysis would result in zero marks for this assignment.

Tools Used Are:

1.Firstly with the help of python code we firstly checked if the given ciphertext is encrypted with CAESAR CIPHER or not.

2. We used python code (attached in answer 6) for finding the frequency of each letter and bi-grams and trigrams analysis in the ciphertext.

3.Then we used table showing letter frequencies in English language text from the lecture slides

Observations:

1. The first thing that came to our mind when we saw the ciphertext was that it could be either Caesar cipher or Substitution Cipher because so many terms were getting repeated.

2.Then we tried all possible shifts with the as in Caesar cipher(shift cipher) we can have utmost 26 shifts we tried out all of them.But unfortunately none of the shifts gave us any sensible text.

3.Now then we moved to frequency analysis to check for substitution cipher.

4.When we did the frequency analysis then found out that the most frequent letter [['y', 0.11009174311926606], ['a', 0.07951070336391437], ['w', 0.0764525993883792], ['m', 0.0764525993883792], ['e', 0.0672782874617737], ['g', 0.04281345565749235], ['s', 0.03669724770642202], ['p', 0.03669724770642202], ['h', 0.03669724770642202], ['i', 0.027522935779816515], ['j', 0.021406727828746176], ['o',0.021406727828746176], ['t', 0.01834862385321101], ['n', 0.01834862385321101], ['u', 0.01834862385321101], ['r', 0.01529051987767584], ['k', 0.01529051987767584], ['v', 0.012232415902140673], ['f', 0.012232415902140673], ['M', 0.009174311926605505], ['x', 0.009174311926605505], ['d', 0.009174311926605505], ['b', 0.0061162079510703364], ['P', 0.0030581039755351682], ['A', 0.0030581039755351682], ['S', 0.0030581039755351682], ['N', 0.0030581039755351682]]

5. From this we can see that the most common letter is 'y'

6.Then we did bigram analysis.There were many bigrams possible the most frequent bigrams frequency are:('m', 'e'): 14 times ('e', 'y'): 9 times ('y', 's'): 8 times ('w', 'a'): 7 times ('a', 'm'): 7 times ('y', 'a'): 7 times ('e', 'w'): 6 times ('w', 'h'): 6 times ('e', 'p'): 5 times ('o', 'y'): 5 times ('m', 'y'): 5 times ('i', 'e'): 4 times

7.Then we did Trigram analysis.There were many trigrams possible the most frequent trigrams frequency are:('m', 'e', 'y'): 8 times ('m', 'e', 'w'): 4 times ('e', 'w', 'a'): 3 times ('i', 'e', 'p'): 3 times ('e', 'p', 'j'): 3 times ('p', 'j', 'o'): 3 times ('j', 'o', 'y'): 3 times ('o', 'y', 's'): 3 times ('e', 'y', 'i'): 3 times ('y', 's', 'y'): 3 times ('w', 'h', 'm'): 3 times ('m', 'y', 's'): 3 times ('y', 'a', 'm'): 3 times ('a', 'm', 'w'): 3 times

8.Now we compare the result of the result of bigrams with the most the most common bigrams in English text.The most common bigrams in english text are following in dressing order of their frequency:- th, he, in, en, nt, re, er, an, ti, es, on, at, se, nd, or, ar, al, te, co, de, to, ra, et, ed, it, sa, em, ro

9.Now we compare the result of the result of trigrams with the most the most common trigrams in English text.The most common trigrams in english text are following in dressing order of their frequency:- the, and, tha, ent, ing, ion, tio, for, nde, has, nce, edt, tis, oft, sth, men

10.Now I saw the most frequent letter in our cipher text we see that it is ‘y’ and we know that most common letters in English alphabets are e,t,,a,o,i,n,s,h,..

11.Therefore we replaced ‘y’->’e’

12.Also through the bigram and trigram analysis we know that the most frequent bigram and trigram occurring in the ciphertext are ‘me’ and ‘mey’.Also we know that most frequent bigram is ‘th’ and trigram is ‘the’.Therefore we replace ‘m’->’t’ and ‘e’->’h’

13.On seeing ‘oe’ here e is already replaced therefore o should be replaced by ‘b’ so as to get a sensible name. ‘o’->’b’.Also we see the word “beeh” here bee are already replaced.Therefore h should be replaced by n. ‘h’->’n’

14.‘p’ is a single letter therefore it should map to ‘p’->‘a’.Now “gne!” ,here n,e are already replaced,therefore ‘g’->’o’. Now “ian” ,here n,a are already replaced,therefore ‘i’->’c’.Now “aee” ,here e is already replaced,therefore ‘a’->’s’.

15.Now “thws” ,here t,h,s are already replaced,therefore ‘w’->’i’.Now “cabes” ,here c,a,e,s are already replaced,therefore ‘b’->’v’.

16.Then “nothinr” ,here n,o,t,h,i,n are already replaced,therefore ‘r’->’g’.Then “intesest” ,here i,n,t,e,s are already replaced,therefore ‘s’->’r’

17.Then “tirst” here i,r,s,t are already replaced therefore ‘t’->’f’.Also “chajber” here c,h,a,b,e,r are already replaced therefore ‘j’->’m’.

18.Now we see that ‘cifher’ here c,i,h,e,r are already replaced therefore ‘f’->’p’. Similarly, “vhich udigts” here h,i,c,d,i,g,t,s are already replaced therefore ‘v’->’w’, ’u’->’d’.Now “wikk” here w,i are already replaced therefore we replace ‘k’->’l’.Now “nsed” here s,e,d are already replaced therefore ‘n’->’u’

19.Also “xou” here o,u are already replaced therefore ‘x’->’y’.Now “doutes” here o,u,t,e,s are already replaced therefore ‘d’->’q’.

20. The mapping we got is as follows:
a→s, b→v, d→q, e→h, f→p, g→o, h→n, i→ c, j→m, k→l, m→t, n→u,
o→b, p→a, r→g, s→r, t→f, u→d, v→w, w→i, x→y, y→e, 8→4, 0→6,
3→9, .→.,"→", ,→, , !→!

21.Therefore this is our cipher text after letter replacement (only letter replacement):-
This is the first chamber of the caves. as you can see, there is nothing of interest in the chamber. some of the later chambers will be more interesting than this one! the code used for this message is a simple substitution cipher in which digits have been shifted by 8 places. the password is "tyRgU03diqq" without the quotes.

22. As the line says "digits have been shifted by 8 places", this infers that the digit 8 must have been

something else, let's say it was 'x', so the line goes "digits have been shifted by x places", now 'x' after shifting by 'x' places becomes 8, which gives the following mathematical expression "(x+x)mod10=8" or "(2*x)mod10=8" on solving we get the x as either '4' or '9'. On applying this substitution to our password, we either have our password as "tyRgU69diqq" or "tyRgU14diqq", after substituting the digits by 4 and 9 places respectively. The later was not accepted at the cave and the former was accepted.

23. So our password is freezed to be "tyRgU69diqq".

Q4 Mapping

10 Points

What is the plaintext space and ciphertext space?
What is the mapping between the elements of plaintext space and the elements of ciphertext space? (Explain in less than 100 words)

The ciphertext space was:
Mewa wa mey twsam iepjoys gt mey ipbya. Pa xgn iph ayy, meysy wa hgmewhr gt whmysyam wh mey iepjoys. Agjy gt mey kpmys iepjoysa vwkk oy jgsy whmysyamwhr meph mewa ghy! Mey iguy nayu tgs mewa jyaapry wa p awjfky anoamwmnmwgh iwfeys wh vewie uwrwma epby oyyh aewtmyu ox 8 fkpiya. Mey fpaavgsu wa "mxSrN03uwdd" vwmegnm mey dngmya.

The mapping is as follows:
a→s, b→v, d→q, e→h, f→p, g→o, h→n, i→ c, j→m, k→l, m→t, n→u,
o→b, p→a, r→g, s→r, t→f, u→d, v→w, w→i, x→y, y→e, 8→4, 0→6,
3→9, .→.,"→", ,→, , !→!

The mapping of "C,L,Q and Z" cannot be inferred from the provided paragraph
The capital letter mappings are same as the small letter mappings.
Space is mapped as space

After using the mapping, the plaintext space has become: (after substitution of alphabets and rotation of digits)

This is the first chamber of the caves. As you can see, there is nothing of interest in the chamber. Some of the later chambers will be more interesting than this one! The code used for this message is a simple substitution cipher in which digits have been shifted by 4 places. The password is "tyRgU69diqq" without the quotes.

Q5 Password
5 Points

What is the final command used to clear this level?

tyRgU69diqq

Q6 Codes
0 Points

Upload any code that you have used to solve this level

Download

SecureSavants__ModernCrypto_Assign1.ipynb

In [ ]:
```
#FREQUENCY ANALYSIS
cipher_text="Mewa wa mey twsam
```

```
iepjoys gt mey ipbya. Pa xgn iph
ayy, meysy wa hgmewhr gt whmysyam
wh mey iepjoys. Agjy gt mey kpmys
iepjoysa vwkk oy jgsy whmysyamwhr
meph mewa ghy! Mey iguy nayu tgs
mewa jyaapry wa p awjfky
anoamwmnmwgh iwfeys wh vewie
uwrwma epby oyyh aewtmyu ox 8
fkpiya. Mey fpaavgsu wa
'mxSrN03uwdd' vwmegnm mey
dngmya."

letter_list=[]
letter_freq_list=[]
count2=0
for l in cipher_text:
    count2=count2+1
for letter_cipher in cipher_text:
    if(letter_cipher in
letter_list):
        continue
    elif(letter_cipher=='.' or
letter_cipher=="'" or
letter_cipher==" " or
letter_cipher.isdigit() or
letter_cipher=="!" or
letter_cipher==","):
        continue
    else:

letter_list.append(letter_cipher)
        temp=[]
        count= count_of_a =
cipher_text.count(letter_cipher)

temp.append(letter_cipher)
        temp.append(count/count2)

letter_freq_list.append(temp)

sorted_list =
sorted(letter_freq_list,
key=lambda y: y[1], reverse=True)
print(sorted_list)
```

```
[['y', 0.11009174311926606], ['a', 0.
```

In [3]:

```
#Bigram Analysis
def
calculate_bigram_frequency(input_stri
    # Remove spaces and convert to
lowercase
    input_string = input_string.repla
", "").lower()

    # Create a dictionary to store th
frequency of each bigram
    bigram_frequency = {}

    # Iterate through the string to
create and count bigrams
    for i in range(len(input_string)
1):
        # Combine consecutive letters
form a bigram
        bigram = (input_string[i],
input_string[i + 1])

        # Update the frequency in the
dictionary
        bigram_frequency[bigram] =
bigram_frequency.get(bigram, 0) + 1

    # Sort the bigrams based on frequ
in descending order
    sorted_bigrams =
sorted(bigram_frequency.items(),
key=lambda x: x[1], reverse=True)

    return sorted_bigrams

# Example usage
input_string = "Mewa wa mey twsam iep
gt mey ipbya. Pa xgn iph ayy, meysy w
hgmewhr gt whmysyam wh mey iepjoys. A
gt mey kpmys iepjoysa vwkk oy jgsy
whmysyamwhr meph mewa ghy! Mey iguy r
tgs mewa jyaapry wa p awjfky anoamwmr
```

```
iwfeys wh vewie uwrwma epby oyyh aewt
ox 8 fkpiya. Mey fpaavgsu wa
'mxSrN03uwdd' vwmegnm mey dngmya."
result =
calculate_bigram_frequency(input_stri

# Display the results
for bigram, frequency in result:
    print(f"{bigram}: {frequency} tim
```

```
('m', 'e'): 14 times
('e', 'y'): 9 times
('y', 's'): 8 times
('w', 'a'): 7 times
('y', 'a'): 7 times
('e', 'w'): 6 times
('w', 'h'): 6 times
('a', 'm'): 5 times
('e', 'p'): 5 times
('o', 'y'): 5 times
('m', 'y'): 5 times
('i', 'e'): 4 times
('s', 'y'): 4 times
('h', 'm'): 4 times
('m', 'w'): 4 times
('p', 'j'): 3 times
('j', 'o'): 3 times
('g', 't'): 3 times
('t', 'm'): 3 times
('y', 'i'): 3 times
('a', '.'): 3 times
('p', 'a'): 3 times
('y', 'w'): 3 times
('g', 's'): 3 times
('w', 'm'): 3 times
('u', 'w'): 3 times
('a', 'w'): 2 times
('t', 'w'): 2 times
('s', 'a'): 2 times
('i', 'p'): 2 times
('p', 'b'): 2 times
('b', 'y'): 2 times
('g', 'n'): 2 times
('p', 'h'): 2 times
('h', 'a'): 2 times
```

```
('a', 'y'): 2 times
('y', 'y'): 2 times
('g', 'm'): 2 times
('h', 'r'): 2 times
('a', 'g'): 2 times
('j', 'y'): 2 times
('k', 'p'): 2 times
('a', 'v'): 2 times
('v', 'w'): 2 times
('g', 'h'): 2 times
('y', 'u'): 2 times
('a', 'a'): 2 times
('a', 'p'): 2 times
('f', 'k'): 2 times
('n', 'm'): 2 times
('a', 'e'): 2 times
('y', 't'): 1 times
('w', 's'): 1 times
('m', 'i'): 1 times
('s', 'g'): 1 times
('.', 'p'): 1 times
('a', 'x'): 1 times
('x', 'g'): 1 times
('n', 'i'): 1 times
('y', ','): 1 times
(',', 'm'): 1 times
('a', 'h'): 1 times
('h', 'g'): 1 times
('r', 'g'): 1 times
('s', '.'): 1 times
('.', 'a'): 1 times
('g', 'j'): 1 times
('y', 'g'): 1 times
('y', 'k'): 1 times
('p', 'm'): 1 times
('s', 'i'): 1 times
('w', 'k'): 1 times
('k', 'k'): 1 times
('k', 'o'): 1 times
('y', 'j'): 1 times
('j', 'g'): 1 times
('r', 'm'): 1 times
('h', 'y'): 1 times
('y', '!'): 1 times
('!', 'm'): 1 times
```

```
('i', 'g'): 1 times
('g', 'u'): 1 times
('u', 'y'): 1 times
('y', 'n'): 1 times
('n', 'a'): 1 times
('u', 't'): 1 times
('t', 'g'): 1 times
('s', 'm'): 1 times
('a', 'j'): 1 times
('p', 'r'): 1 times
('r', 'y'): 1 times
('w', 'j'): 1 times
('j', 'f'): 1 times
('k', 'y'): 1 times
('a', 'n'): 1 times
('n', 'o'): 1 times
('o', 'a'): 1 times
('m', 'n'): 1 times
('w', 'g'): 1 times
('h', 'i'): 1 times
('i', 'w'): 1 times
('w', 'f'): 1 times
('f', 'e'): 1 times
('s', 'w'): 1 times
('h', 'v'): 1 times
('v', 'e'): 1 times
('w', 'i'): 1 times
('e', 'u'): 1 times
('w', 'r'): 1 times
('r', 'w'): 1 times
('m', 'a'): 1 times
('y', 'o'): 1 times
('y', 'h'): 1 times
('w', 't'): 1 times
('u', 'o'): 1 times
('o', 'x'): 1 times
('x', '8'): 1 times
('8', 'f'): 1 times
('p', 'i'): 1 times
('i', 'y'): 1 times
('.', 'm'): 1 times
('y', 'f'): 1 times
('f', 'p'): 1 times
('v', 'g'): 1 times
('s', 'u'): 1 times
```

```
('a', "'"): 1 times
("'", 'm'): 1 times
('m', 'x'): 1 times
('x', 's'): 1 times
('s', 'r'): 1 times
('r', 'n'): 1 times
('n', '0'): 1 times
('0', '3'): 1 times
('3', 'u'): 1 times
('w', 'd'): 1 times
('d', 'd'): 1 times
('d', "'"): 1 times
("'", 'v'): 1 times
('e', 'g'): 1 times
('m', 'm'): 1 times
('y', 'd'): 1 times
('d', 'n'): 1 times
('n', 'g'): 1 times
```

In [4]:

```
def
calculate_trigram_frequency(input_str
    # Remove spaces and convert to
lowercase
    input_string = input_string.repla
", "").lower()

    # Create a dictionary to store th
frequency of each trigram
    trigram_frequency = {}

    # Iterate through the string to c
and count trigrams
    for i in range(len(input_string)
        # Combine consecutive letters
form a trigram
        trigram = (input_string[i],
input_string[i + 1], input_string[i +

        # Update the frequency in the
dictionary
        trigram_frequency[trigram] =
trigram_frequency.get(trigram, 0) + 1

    # Sort the trigrams based on freq
```

```
in descending order
    sorted_trigrams =
sorted(trigram_frequency.items(),
key=lambda x: x[1], reverse=True)

    return sorted_trigrams

# Example usage
input_string = "Mewa wa mey twsam iep
gt mey ipbya. Pa xgn iph ayy, meysy w
hgmewhr gt whmysyam wh mey iepjoys. A
gt mey kpmys iepjoysa vwkk oy jgsy
whmysyamwhr meph mewa ghy! Mey iguy r
tgs mewa jyaapry wa p awjfky anoamwmn
iwfeys wh vewie uwrwma epby oyyh aewt
ox 8 fkpiya. Mey fpaavgsu wa 'mxSrN03
vwmegnm mey dngmya."
result =
calculate_trigram_frequency(input_str

# Display the results
for trigram, frequency in result:
    print(f"{trigram}: {frequency} ti
```

```
('m', 'e', 'y'): 8 times
('m', 'e', 'w'): 4 times
('e', 'w', 'a'): 3 times
('i', 'e', 'p'): 3 times
('e', 'p', 'j'): 3 times
('p', 'j', 'o'): 3 times
('j', 'o', 'y'): 3 times
('o', 'y', 's'): 3 times
('e', 'y', 'i'): 3 times
('y', 'a', '.'): 3 times
('y', 's', 'y'): 3 times
('w', 'h', 'm'): 3 times
('m', 'y', 's'): 3 times
('a', 'm', 'w'): 3 times
('g', 't', 'm'): 2 times
('t', 'm', 'e'): 2 times
('p', 'b', 'y'): 2 times
('e', 'y', 's'): 2 times
('s', 'y', 'w'): 2 times
('y', 'w', 'a'): 2 times
('w', 'h', 'r'): 2 times
```

```
('h', 'm', 'y'): 2 times
('s', 'y', 'a'): 2 times
('y', 'a', 'm'): 2 times
('m', 'w', 'h'): 2 times
('h', 'm', 'e'): 2 times
('w', 'a', 'w'): 1 times
('a', 'w', 'a'): 1 times
('w', 'a', 'm'): 1 times
('a', 'm', 'e'): 1 times
('e', 'y', 't'): 1 times
('y', 't', 'w'): 1 times
('t', 'w', 's'): 1 times
('w', 's', 'a'): 1 times
('s', 'a', 'm'): 1 times
('a', 'm', 'i'): 1 times
('m', 'i', 'e'): 1 times
('y', 's', 'g'): 1 times
('s', 'g', 't'): 1 times
('y', 'i', 'p'): 1 times
('i', 'p', 'b'): 1 times
('b', 'y', 'a'): 1 times
('a', '.', 'p'): 1 times
('.', 'p', 'a'): 1 times
('p', 'a', 'x'): 1 times
('a', 'x', 'g'): 1 times
('x', 'g', 'n'): 1 times
('g', 'n', 'i'): 1 times
('n', 'i', 'p'): 1 times
('i', 'p', 'h'): 1 times
('p', 'h', 'a'): 1 times
('h', 'a', 'y'): 1 times
('a', 'y', 'y'): 1 times
('y', 'y', ','): 1 times
('y', ',', 'm'): 1 times
(',', 'm', 'e'): 1 times
('w', 'a', 'h'): 1 times
('a', 'h', 'g'): 1 times
('h', 'g', 'm'): 1 times
('g', 'm', 'e'): 1 times
('e', 'w', 'h'): 1 times
('h', 'r', 'g'): 1 times
('r', 'g', 't'): 1 times
('g', 't', 'w'): 1 times
('t', 'w', 'h'): 1 times
('y', 'i', 'e'): 1 times
```

```
('y', 's', '.'): 1 times
('s', '.', 'a'): 1 times
('.', 'a', 'g'): 1 times
('a', 'g', 'j'): 1 times
('g', 'j', 'y'): 1 times
('j', 'y', 'g'): 1 times
('y', 'g', 't'): 1 times
('e', 'y', 'k'): 1 times
('y', 'k', 'p'): 1 times
('k', 'p', 'm'): 1 times
('p', 'm', 'y'): 1 times
('y', 's', 'i'): 1 times
('s', 'i', 'e'): 1 times
('y', 's', 'a'): 1 times
('s', 'a', 'v'): 1 times
('a', 'v', 'w'): 1 times
('v', 'w', 'k'): 1 times
('w', 'k', 'k'): 1 times
('k', 'k', 'o'): 1 times
('k', 'o', 'y'): 1 times
('o', 'y', 'j'): 1 times
('y', 'j', 'g'): 1 times
('j', 'g', 's'): 1 times
('g', 's', 'y'): 1 times
('y', 'w', 'h'): 1 times
('h', 'r', 'm'): 1 times
('r', 'm', 'e'): 1 times
('m', 'e', 'p'): 1 times
('e', 'p', 'h'): 1 times
('p', 'h', 'm'): 1 times
('w', 'a', 'g'): 1 times
('a', 'g', 'h'): 1 times
('g', 'h', 'y'): 1 times
('h', 'y', '!'): 1 times
('y', '!', 'm'): 1 times
('!', 'm', 'e'): 1 times
('y', 'i', 'g'): 1 times
('i', 'g', 'u'): 1 times
('g', 'u', 'y'): 1 times
('u', 'y', 'n'): 1 times
('y', 'n', 'a'): 1 times
('n', 'a', 'y'): 1 times
('a', 'y', 'u'): 1 times
('y', 'u', 't'): 1 times
('u', 't', 'g'): 1 times
```

```
('t', 'g', 's'): 1 times
('g', 's', 'm'): 1 times
('s', 'm', 'e'): 1 times
('w', 'a', 'j'): 1 times
('a', 'j', 'y'): 1 times
('j', 'y', 'a'): 1 times
('y', 'a', 'a'): 1 times
('a', 'a', 'p'): 1 times
('a', 'p', 'r'): 1 times
('p', 'r', 'y'): 1 times
('r', 'y', 'w'): 1 times
('w', 'a', 'p'): 1 times
('a', 'p', 'a'): 1 times
('p', 'a', 'w'): 1 times
('a', 'w', 'j'): 1 times
('w', 'j', 'f'): 1 times
('j', 'f', 'k'): 1 times
('f', 'k', 'y'): 1 times
('k', 'y', 'a'): 1 times
('y', 'a', 'n'): 1 times
('a', 'n', 'o'): 1 times
('n', 'o', 'a'): 1 times
('o', 'a', 'm'): 1 times
('m', 'w', 'm'): 1 times
('w', 'm', 'n'): 1 times
('m', 'n', 'm'): 1 times
('n', 'm', 'w'): 1 times
('m', 'w', 'g'): 1 times
('w', 'g', 'h'): 1 times
('g', 'h', 'i'): 1 times
('h', 'i', 'w'): 1 times
('i', 'w', 'f'): 1 times
('w', 'f', 'e'): 1 times
('f', 'e', 'y'): 1 times
('y', 's', 'w'): 1 times
('s', 'w', 'h'): 1 times
('w', 'h', 'v'): 1 times
('h', 'v', 'e'): 1 times
('v', 'e', 'w'): 1 times
('e', 'w', 'i'): 1 times
('w', 'i', 'e'): 1 times
('i', 'e', 'u'): 1 times
('e', 'u', 'w'): 1 times
('u', 'w', 'r'): 1 times
('w', 'r', 'w'): 1 times
```

```
('r', 'w', 'm'): 1 times
('w', 'm', 'a'): 1 times
('m', 'a', 'e'): 1 times
('a', 'e', 'p'): 1 times
('e', 'p', 'b'): 1 times
('b', 'y', 'o'): 1 times
('y', 'o', 'y'): 1 times
('o', 'y', 'y'): 1 times
('y', 'y', 'h'): 1 times
('y', 'h', 'a'): 1 times
('h', 'a', 'e'): 1 times
('a', 'e', 'w'): 1 times
('e', 'w', 't'): 1 times
('w', 't', 'm'): 1 times
('t', 'm', 'y'): 1 times
('m', 'y', 'u'): 1 times
('y', 'u', 'o'): 1 times
('u', 'o', 'x'): 1 times
('o', 'x', '8'): 1 times
('x', '8', 'f'): 1 times
('8', 'f', 'k'): 1 times
('f', 'k', 'p'): 1 times
('k', 'p', 'i'): 1 times
('p', 'i', 'y'): 1 times
('i', 'y', 'a'): 1 times
('a', '.', 'm'): 1 times
('.', 'm', 'e'): 1 times
('e', 'y', 'f'): 1 times
('y', 'f', 'p'): 1 times
('f', 'p', 'a'): 1 times
('p', 'a', 'a'): 1 times
('a', 'a', 'v'): 1 times
('a', 'v', 'g'): 1 times
('v', 'g', 's'): 1 times
('g', 's', 'u'): 1 times
('s', 'u', 'w'): 1 times
('u', 'w', 'a'): 1 times
('w', 'a', "'"): 1 times
('a', "'", 'm'): 1 times
("'", 'm', 'x'): 1 times
('m', 'x', 's'): 1 times
('x', 's', 'r'): 1 times
('s', 'r', 'n'): 1 times
('r', 'n', '0'): 1 times
('n', '0', '3'): 1 times
```

```
('0', '3', 'u'): 1 times
('3', 'u', 'w'): 1 times
('u', 'w', 'd'): 1 times
('w', 'd', 'd'): 1 times
('d', 'd', "'"): 1 times
('d', "'", 'v'): 1 times
("'", 'v', 'w'): 1 times
('v', 'w', 'm'): 1 times
('w', 'm', 'e'): 1 times
('m', 'e', 'g'): 1 times
('e', 'g', 'n'): 1 times
('g', 'n', 'm'): 1 times
('n', 'm', 'm'): 1 times
('m', 'm', 'e'): 1 times
('e', 'y', 'd'): 1 times
('y', 'd', 'n'): 1 times
('d', 'n', 'g'): 1 times
('n', 'g', 'm'): 1 times
('g', 'm', 'y'): 1 times
('m', 'y', 'a'): 1 times
```

In [11]:

```
a="Mewa wa mey twsam iepjoys gt mey i
xgn iph ayy, meysy wa hgmewhr gt whmy
mey iepjoys. Agjy gt mey kpmys iepjoy
oy jgsy whmysyamwhr meph mewa ghy! Me
nayu tgs mewa jyaapry wa p awjfky
anoamwmnmwgh iwfeys wh vewie uwrwma e
aewtmyu ox 8 fkpiya. Mey fpaavgsu wa
*mxSrN03uwdd* vwmegnm mey dngmya."
e="Mewa wa mey twsam iepjoys gt mey i
xgn iph ayy, meysy wa hgmewhr gt whmy
mey iepjoys. Agjy gt mey kpmys iepjoy
oy jgsy whmysyamwhr meph mewa ghy! Me
nayu tgs mewa jyaapry wa p awjfky
anoamwmnmwgh iwfeys wh vewie uwrwma e
aewtmyu ox 8 fkpiya. Mey fpaavgsu wa
'"'+"mxSrN03uwdd"+'"'+" vwmegnm mey d
a=list(a)
b=[]
for i in range(97,123):
    g=[]
    g.append(chr(i))
    g.append(0)
    b.append(g)
```

```
#print(b)

for k in range(26):
    for i in range(len(a)):
        if(a[i]==chr(k+97)):
            b[k][1]=b[k][1]+1

dd = sorted(b, key=lambda x: x[1])
dd=dd[::-1]



f=
{'a':'s','b':'v','d':'q','e':'h','f':
mapping','l':'no mapping','q':'no
mapping','z':'no mapping'}

print(" ")
print("The Mapping")
for k in range(26):
    print(chr(k+97)+" -> "+f[chr(k+97


print(" ")
print("The Ciphertext: ")
print(e)
print(" ")

print(" ")
print("The Message: ")
print(" ")

for i in range(len(a)):
    if(ord(a[i])>=65 and ord(a[i])<=9

print(str(chr(ord(f[chr(ord(a[i])+32)
    elif(a[i]==' ' or a[i]=='.'or a[i
or a[i]=='!'):
        print(a[i],end="")
    elif(a[i]=="*"):
        print('"',end="")
    elif(ord(a[i])>=48 and ord(a[i])<
        zz=int(a[i])
        zz=(zz-4)%10
        print(str(zz),end="")
    else:
        print(str(f[a[i]]),end="")
```

```
The Mapping
a -> s
b -> v
c -> no mapping
d -> q
e -> h
f -> p
g -> o
h -> n
i -> c
j -> m
k -> l
l -> no mapping
m -> t
n -> u
o -> b
p -> a
q -> no mapping
r -> g
s -> r
t -> f
u -> d
v -> w
w -> i
x -> y
y -> e
z -> no mapping

The Ciphertext:
Mewa wa mey twsam iepjoys gt mey ipby


The Message:

This is the first chamber of the cave
```

Q7 Team Name
0 Points

Enter your Team Name

SecureSavants

Assignment 1

● Graded

Select each question to review feedback and grading details.

Group

VISHAL KUMAR
Souvik Mukherjee
Vikrant Chauhan
View or edit group

Total Points

50 / 50 pts

Question 1	
Commands	**5** / 5 pts
Question 2	
Cryptosystem	**5** / 5 pts
Question 3	
Analysis	**25** / 25 pts

Question 4

Mapping **10** / 10 pts

Question 5

Password **5** / 5 pts

Question 6

Codes **0** / 0 pts

Question 7

Team Name **0** / 0 pts