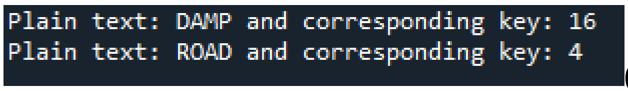
Question 1)

We have key space as 26 because of English alphabet size. Since key space is small, exhaustive search (brute force) can be used for decryption. In the python file, I tried all possible key values [0,25] and recorded corresponding decrypted plain texts. Then, I used NLTK (Natural Language Tool Kit) to identify possible English words in these plain texts.

There are 2 plain texts: “DAMP” and “ROAD”.

Their encryption keys are: 16 and 4



```
Plain text: DAMP and corresponding key: 16
Plain text: ROAD and corresponding key: 4
```

(CS411_HW1_q1.py output)

Corresponding codes can be found in the CS411_HW1_q1.py

Question 2)

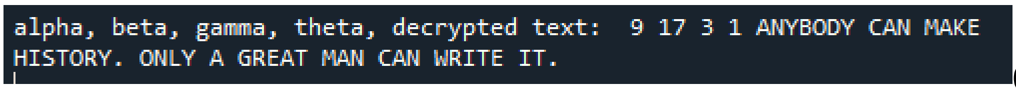
Firstly, we need to find most frequent letter in the cipher text as we already know the most frequent letter in plain text. If we know a pair of letters, we can use this information to reduce our key space. We have “R” as the most frequent letter in the cipher text. Now, we have the pair (plaintext - ciphertext) of A(0)-R(17). If we put these values into Affine Cipher formula, we can have possible alpha-beta pairs to identify our key.

Affine Cipher Encryption:

$$\text{Encryption: } E_k(x) = y = \alpha \cdot x + \beta \bmod 26$$

We have following equation by using A(0)-R(17) pair: 17 = β mod (26). Then, we can say that our β is 17.

We have 12 possible α values which have inverse in the mod 26. Then, we can say that we have key space as 12. Now, we can try exhaustive search to find our key.



```
alpha, beta, gamma, theta, decrypted text:  9 17 3 1 ANYBODY CAN MAKE
HISTORY. ONLY A GREAT MAN CAN WRITE IT.
```

(q2.py output)

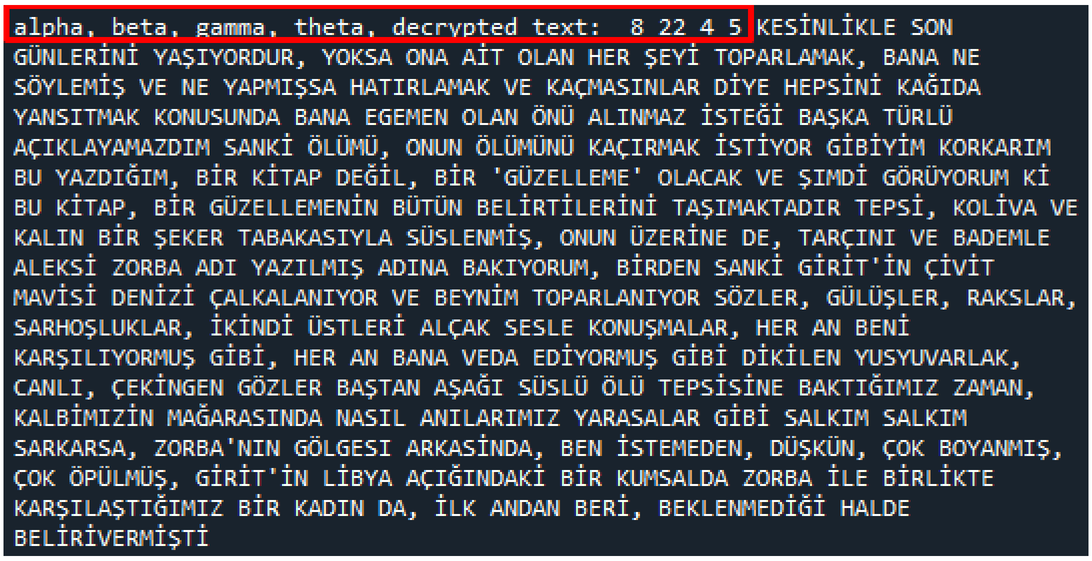
Corresponding codes can be found in the CS411_HW1_q2.py

Question 3)

Firstly, we need to find most frequent letter in the cipher text as we already know the most frequent letter in plain text. I have pair of plaintext and ciphertext A(0)-Ş(22). This is clear that we have β as 22. However, this time I implemented more general algorithm to solve this decryption. I checked all possible alpha – beta pairs that are ensure the below equation:

$$\text{Encryption: } E_k(x) = y = \alpha \cdot x + \beta \bmod 26$$

We had 30 possible pairs (key space). I used exhaustive search (brute force) to find correct key. After trying all possible keys, I found the below result:



```
alpha, beta, gamma, theta, decrypted text:  8 22 4 5 KESİNLİKLE SON
GÜNLERİNİ YAŞIYORDUR, YOKSA ONA AİT OLAN HER ŞEYİ TOPARLAMAK, BANA NE
SÖYLEMİŞ VE NE YAPMIŞSA HATIRLAMAK VE KAÇMASINLAR DİYE HEPSİNİ KAĞIDA
YANSITMAK KONUSUNDA BANA EGEMEN OLAN ÖNÜ ALINMAZ İSTEĞİ BAŞKA TÜRLÜ
AÇIKLAYAMAZDIM SANKİ ÖLÜMÜ, ONUN ÖLÜMÜNÜ KAÇIRMAK İSTİYOR GİBİYİM KORKARIM
BU YAZDIĞIM, BİR KİTAP DEĞİL, BİR 'GÜZELLEME' OLACAK VE ŞIMDİ GÖRÜYORUM Kİ
BU KİTAP, BİR GÜZELLEMENİN BÜTÜN BELİRTİLERİNİ TAŞIMAKTADIR TEPSİ, KOLİVA VE
KALIN BİR ŞEKER TABAKASIYLA SÜSLENMİŞ, ONUN ÜZERİNE DE, TARÇINI VE BADEMLE
ALEKSİ ZORBA ADI YAZILMIŞ ADINA BAKIYORUM, BİRDEN SANKİ GİRİT'İN ÇİVİT
MAVİSİ DENİZİ ÇALKALANIYOR VE BEYNİM TOPARLANIYOR SÖZLER, GÜLÜŞLER, RAKSLAR,
SARHOŞLUKLAR, İKİNDİ ÜSTLERİ ALÇAK SESLE KONUŞMALAR, HER AN BENİ
KARŞILIYORMUŞ GİBİ, HER AN BANA VEDA EDİYORMUŞ GİBİ DİKİLEN YUSYUVARLAK,
CANLI, ÇEKİNGEN GÖZLER BAŞTAN AŞAĞI SÜSLÜ ÖLÜ TEPSİSİNE BAKTIĞIMIZ ZAMAN,
KALBİMİZİN MAĞARASINDA NASIL ANILARIMIZ YARASALAR GİBİ SALKIM SALKIM
SARKARSA, ZORBA'NIN GÖLGESI ARKASİNDA, BEN İSTEMEDEN, DÜŞKÜN, ÇOK BOYANMIŞ,
ÇOK ÖPÜLMÜŞ, GİRİT'İN LİBYA AÇIĞINDAKİ BİR KUMSALDA ZORBA İLE BİRLİKTE
KARŞILAŞTIĞIMIZ BİR KADIN DA, İLK ANDAN BERİ, BEKLENMEDİĞİ HALDE
BELİRİVERMİŞTİ
```

Finally, I sent this plain text to the server to be ensure that I have correct plaintext. I got the following response from server: **Congrats!**

Corresponding codes can be found in the affine_client.py

Question 4)

Since we choose our key as uniformly randomly (total randomness without any predictability and biasedness), any possible ciphertext letter β can be encrypted by any possible plaintext letter α.

We have the following equation for the probability of ciphertext letter β:

$$\underbrace{\sum_{i=0}^{25} P(\alpha_i)}_{1} \cdot P(k) = P(\beta)$$

↳ choosing right key prob

$$1 \cdot P(k) = P(\beta)$$

↓

$$\boxed{\frac{1}{26} = P(\beta)}$$

We can see that we do not care about the probability of letter α, as the key chosen uniformly randomly.

Question 5)

Firstly, we need to find number of possible bigrams. We have 28 choices for the first character, and we have also 28 choices for the second character as they are independent from each other.

This shows that we have 28 * 28 = **784 different bigrams. This is our mod.**

Now, we have the following Affine Cipher Encryption equation:

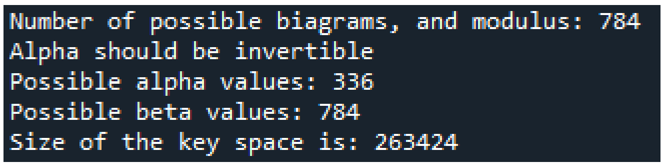
$$\underline{\text{Encryption}}: E_k(x) = y = \alpha \cdot x + \beta \bmod \boxed{\mathbf{784}}$$

Our alpha should have inverse in the mod 784. In other words, alphas should be relatively prime with 784.

Number of alphas that are relatively prime with 784 equals to 336. We do not have any restriction for the beta values. Therefore, we have 784 different options for beta. This ends up with having 336 * 784 = **263424 key space.**

<u>Modulus:</u> 784

<u>Size of the Key Space:</u> 263424



```
Number of possible biagrams, and modulus: 784
Alpha should be invertible
Possible alpha values: 336
Possible beta values: 784
Size of the key space is: 263424
```

(CS411_HW1_q5.py output)

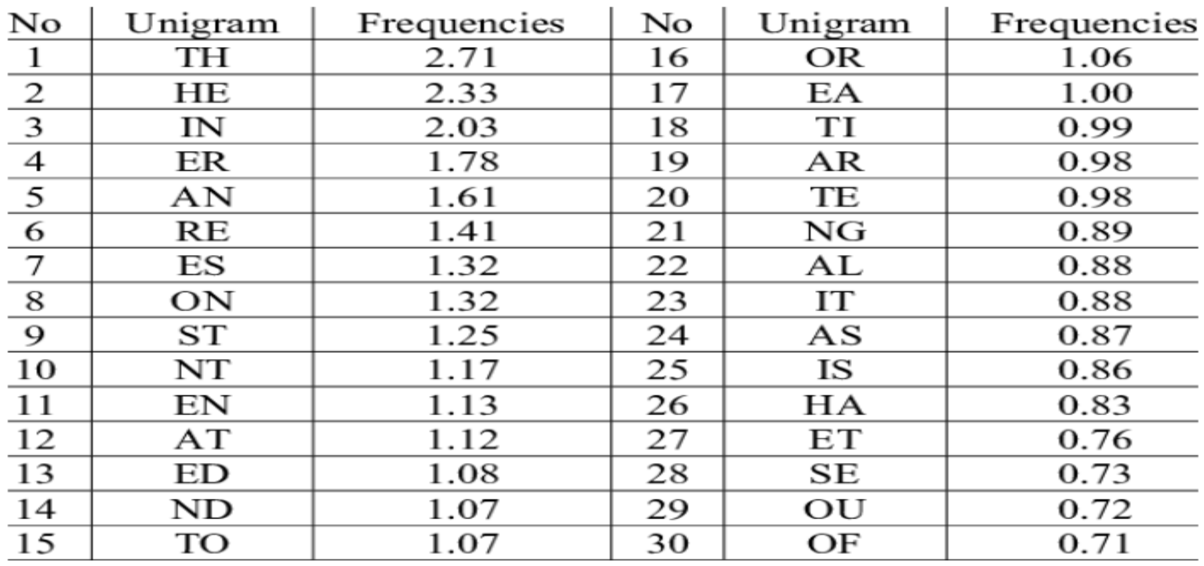
Corresponding codes can be found in the CS411_HW1_q5_q7.py

Question 6)

NO, it does not secure against letter frequency analysis. I will look from two different ways to show that it is not secured.

Firstly, we know that observing individual letters have some different frequencies. For instance, ‘e, t, a’ have higher probability compared to other letters. Thus, I can say that combinations of ‘ee’, ‘et’, ‘ea’… would be more observable in the plain text if the language statistics allow that. Thus, if we analyse most frequent bigrams in the cipher text, we can try to map frequent bigrams in cipher text to these combinations.

Secondly, languages are not formed by randomly chosen characters. There should be also a frequency table for bigrams of all languages. I found the following table by searching on the internet.



No	Unigram	Frequencies	No	Unigram	Frequencies
1	TH	2.71	16	OR	1.06
2	HE	2.33	17	EA	1.00
3	IN	2.03	18	TI	0.99
4	ER	1.78	19	AR	0.98
5	AN	1.61	20	TE	0.98
6	RE	1.41	21	NG	0.89
7	ES	1.32	22	AL	0.88
8	ON	1.32	23	IT	0.88
9	ST	1.25	24	AS	0.87
10	NT	1.17	25	IS	0.86
11	EN	1.13	26	HA	0.83
12	AT	1.12	27	ET	0.76
13	ED	1.08	28	SE	0.73
14	ND	1.07	29	OU	0.72
15	TO	1.07	30	OF	0.71

(Taken: https://www.researchgate.net/figure/English-bigram-frequencies-in-percent_tbl2_343699517)

We can use this table as follows: We can find the most frequent bigram in the cipher text. Then, we can try to map this bigram with the most frequent bigrams in the table.

For instance, if we had ‘AL’ as the most frequent bigram in the cipher text. We can try to use the pair (plaintext - ciphertext) of ‘TH’ and ‘AL’ to find beta and gamma values as we did in the 7. question.

Question 7)

We know that our plain text is not a multiple of 2 thanks to Hint 2. Therefore, we know that plain text is padded with one ‘X’, and we also know that plain text normally ends with ‘.’. Then, we can say that we have pair (plaintext - ciphertext) of ‘.X’(751) and ‘BE’(32).

```
Last two characters of plaintext .X and corresponding code 751
Last two characters of ciphertext BE and corresponding code 32
```

Using this pair, we have following equation:

$$\text{Encryption: } E_k(x) = \boxed{32} = \alpha \boxed{751} + \beta \bmod \boxed{784}$$

Using this equation, we can find the possible alpha – beta pairs. We have 336 possible pairs. We can use brute force to decrypt our cipher text as we have small size of key space.

I divided the cipher text into bigrams to decrypt each bigram one by one. Again, I used NLTK to reduce printed outputs. My result can be seen below:

```
alpha 121, beta 105, gamma 473, theta 511, decrypted text IT DOES NOT DO TO
DWELL ON DREAMS AND FORGET TO LIVE.X:
```

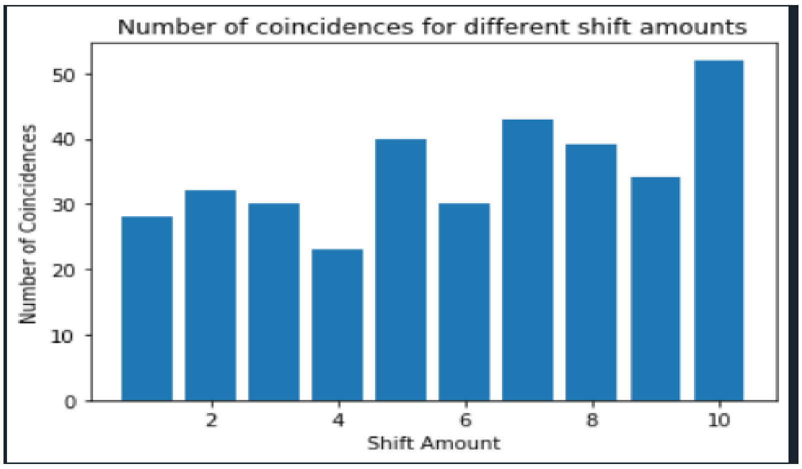
Since, X at the end is additional we can exclude it and obtain following result:

Plaintext: “IT DOES NOT DO TO DWELL ON DREAMS AND FORGET TO LIVE.”

Corresponding codes can be found in the CS411_HW1_q5_q7.py

Question Bonus)

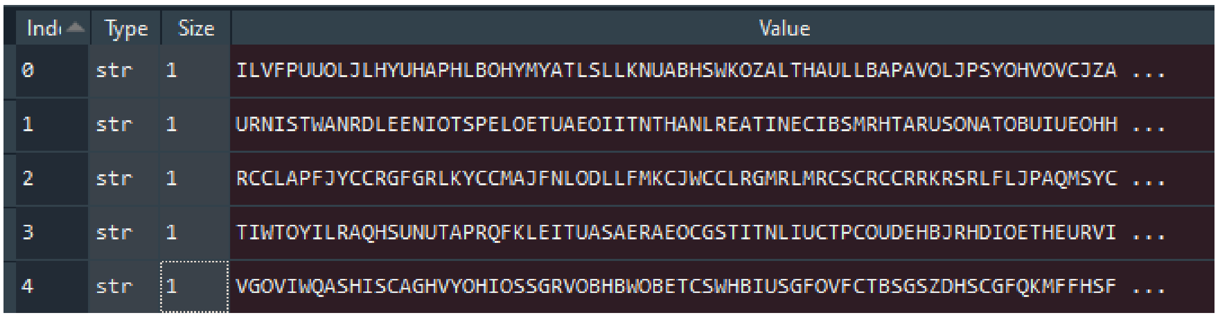
We know that frequency analysis of letters can help to decrypt the cipher text from lectures. To make a frequency analysis, we need to find **length of the key**. If we can find the key length, we can divide cipher text into sub cipher texts to analyse them individually. To find key length, we need to shift cipher text one by one and observe the number of coincidences of letters. Then, the key length is probably equal to shift amount that causes most coincidences. Therefore, I started with observing number of coincidences for different shift amounts up to 10. For counting the coincidences, I removed the non-alphabetic characters from cipher text.



Shift amounts (5, 7, 10) are giving the highest number of coincidences. Shift amount (10) gives the most coincidences. However, I will not start with assuming key length as 10 because making frequency analysis for 10 sub ciphertext can be time consuming for me 😃 and Hoca said look for key length less than 10. Therefore, I will start with the key length as 5.

Assuming key length 5

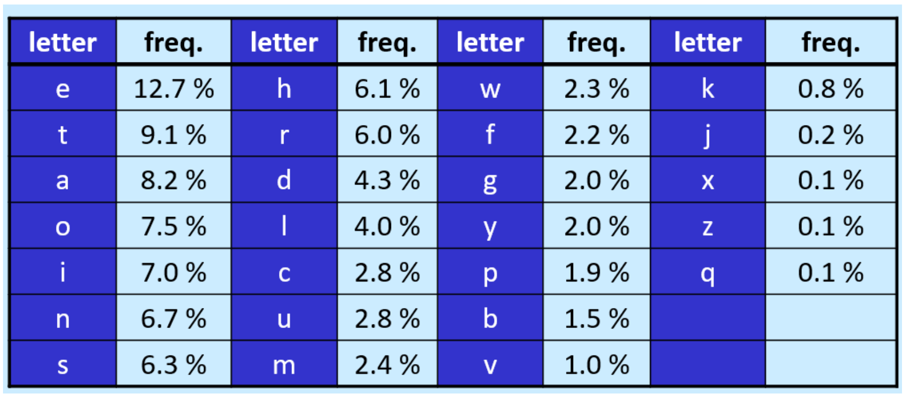
We need to divide our ciphertext into sub ciphertexts as following. Taking [1,6, 11…] letters for first sub cipher text, taking [2, 7, 12, …] for the second sub cipher text and so on. In below we can see all sub ciphertexts.



Indi	Type	Size	Value
0	str	1	ILVFPUUOLJLHYUHAPHLBOHYMYATLSLLKNUABHSWKOZALTHAULLBAPAVOLJPSYOHVOVCJZA ...
1	str	1	URNISTWANRDLEENIOTSPELOETUAEOIITNTHANLREATINECIBSMRHTARUSONATOBUIUEOHH ...
2	str	1	RCCLAPFJYCCRGFGRLKYCCMAJFNLODLLFMKCJWCCLRGMRLMRCSCRCCRRKRSRLFLJPAQMSYC ...
3	str	1	TIWTOYILRAQHSUNUTAPRQFKLEITUASAERAEOCGSTITNLIUCTPCOUDEHBJRHDIOETHEURVI ...
4	str	1	VGOVIWQASHISCAGHVYOHIOSSGRVOBHBWOBETCSWHBIUSGFOVFCTBSGSZDHSCGFQKMFFHSF ...

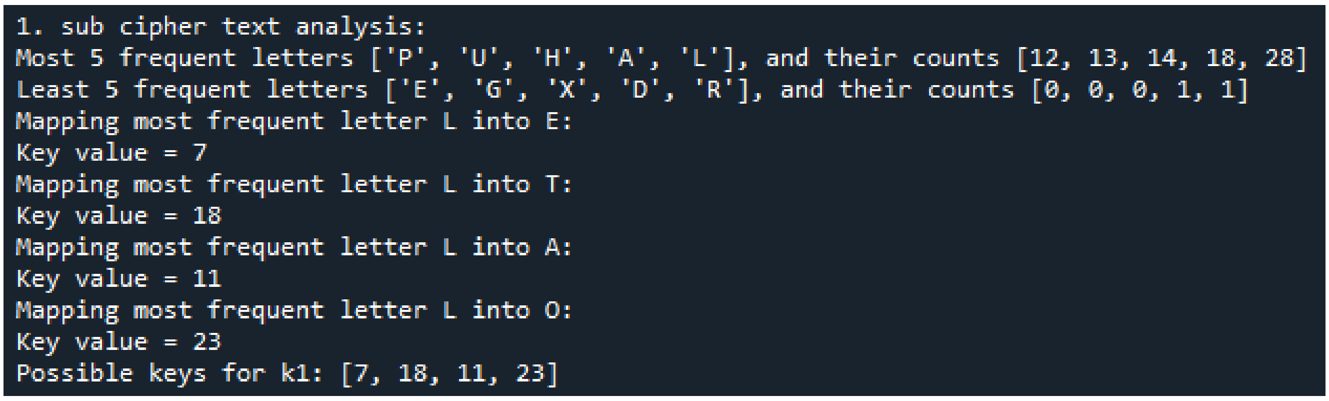
After obtaining sub cipher texts, I recorded the counts of each letter in a dictionary for each sub cipher texts. Then, I tried to do frequency analysis based on these records. For simplicity, I looked for 5 most and least frequent letters in the cipher texts for frequency analysing.

I will try to match letters based on this table from the lecture notes:



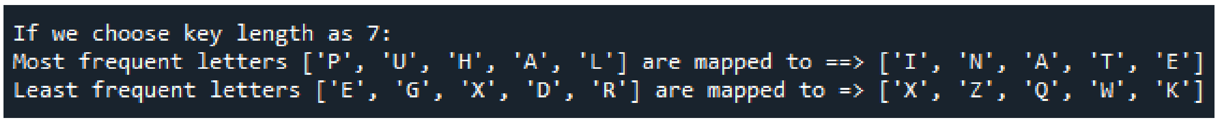
letter	freq.	letter	freq.	letter	freq.	letter	freq.
e	12.7 %	h	6.1 %	w	2.3 %	k	0.8 %
t	9.1 %	r	6.0 %	f	2.2 %	j	0.2 %
a	8.2 %	d	4.3 %	g	2.0 %	x	0.1 %
o	7.5 %	l	4.0 %	y	2.0 %	z	0.1 %
i	7.0 %	c	2.8 %	p	1.9 %	q	0.1 %
n	6.7 %	u	2.8 %	b	1.5 %		
s	6.3 %	m	2.4 %	v	1.0 %		

Analysing 1. Sub Ciphertext:



```
1. sub cipher text analysis:
Most 5 frequent letters ['P', 'U', 'H', 'A', 'L'], and their counts [12, 13, 14, 18, 28]
Least 5 frequent letters ['E', 'G', 'X', 'D', 'R'], and their counts [0, 0, 0, 1, 1]
Mapping most frequent letter L into E:
Key value = 7
Mapping most frequent letter L into T:
Key value = 18
Mapping most frequent letter L into A:
Key value = 11
Mapping most frequent letter L into O:
Key value = 23
Possible keys for k1: [7, 18, 11, 23]
```

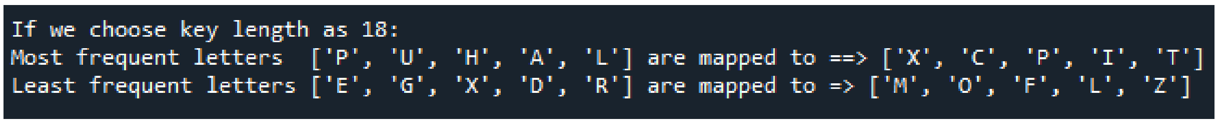
Let’s start with k1 = 7:


```
If we choose key length as 7:
Most frequent letters ['P', 'U', 'H', 'A', 'L'] are mapped to ==> ['I', 'N', 'A', 'T', 'E']
Least frequent letters ['E', 'G', 'X', 'D', 'R'] are mapped to => ['X', 'Z', 'Q', 'W', 'K']
```

This looks consistent as the most frequent letters are matched with the English most frequent letters. Same for the least frequent letters.

Therefore, **k1 = 7** is a good choice for key. To be safe, lets try another possible key for k1.

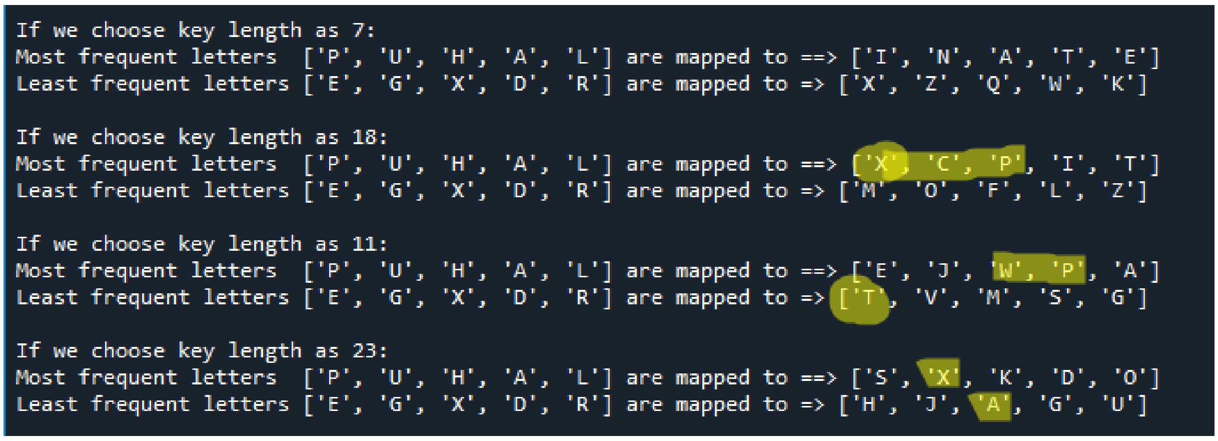
Trying k1 = 18:


```
If we choose key length as 18:
Most frequent letters  ['P', 'U', 'H', 'A', 'L'] are mapped to ==> ['X', 'C', 'P', 'I', 'T']
Least frequent letters ['E', 'G', 'X', 'D', 'R'] are mapped to => ['M', 'O', 'F', 'L', 'Z']
```

It does not look correct key because we have [‘X’, ‘C’, ‘P’] in the most frequent letters. However, these are not true for English language statistics table.

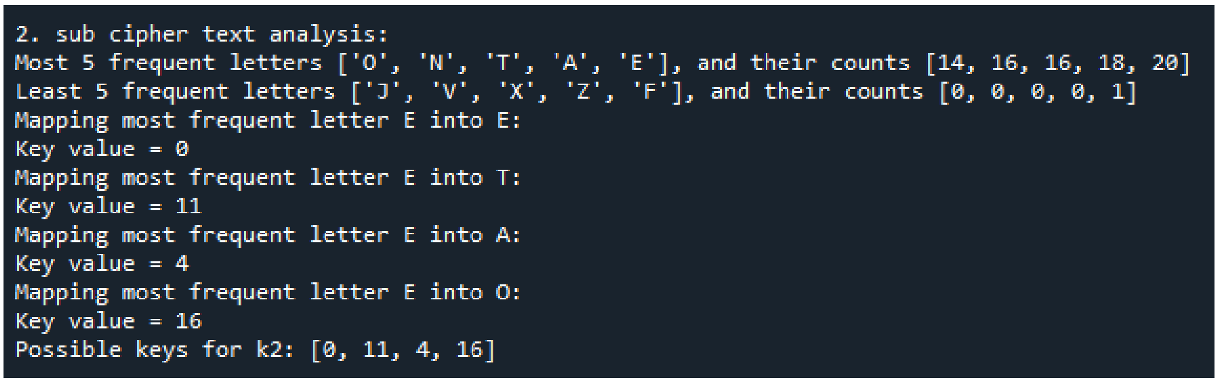
Therefore, k1 = 18 **is not** a good choice for key.

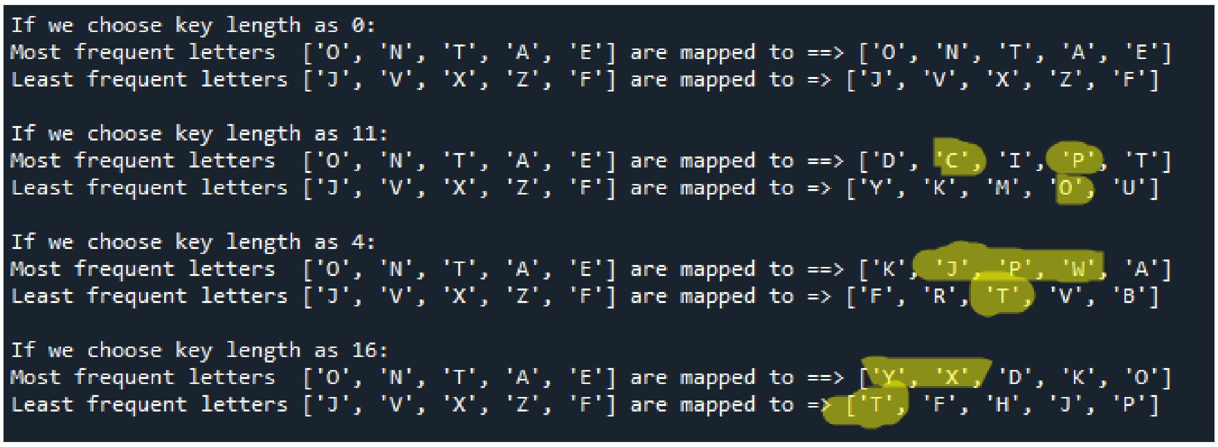
You can see all these possible keys’ result in the below:


```
If we choose key length as 7:
Most frequent letters  ['P', 'U', 'H', 'A', 'L'] are mapped to ==> ['I', 'N', 'A', 'T', 'E']
Least frequent letters ['E', 'G', 'X', 'D', 'R'] are mapped to => ['X', 'Z', 'Q', 'W', 'K']

If we choose key length as 18:
Most frequent letters  ['P', 'U', 'H', 'A', 'L'] are mapped to ==> ['X', 'C', 'P', 'I', 'T']
Least frequent letters ['E', 'G', 'X', 'D', 'R'] are mapped to => ['M', 'O', 'F', 'L', 'Z']

If we choose key length as 11:
Most frequent letters  ['P', 'U', 'H', 'A', 'L'] are mapped to ==> ['E', 'J', 'W', 'P', 'A']
Least frequent letters ['E', 'G', 'X', 'D', 'R'] are mapped to => ['T', 'V', 'M', 'S', 'G']

If we choose key length as 23:
Most frequent letters  ['P', 'U', 'H', 'A', 'L'] are mapped to ==> ['S', 'X', 'K', 'D', 'O']
Least frequent letters ['E', 'G', 'X', 'D', 'R'] are mapped to => ['H', 'J', 'A', 'G', 'U']
```

Most promising one is **k1 = 7** as I explained before. I highlighted obvious inconsistencies.

Analysing 2. Sub Ciphertext:

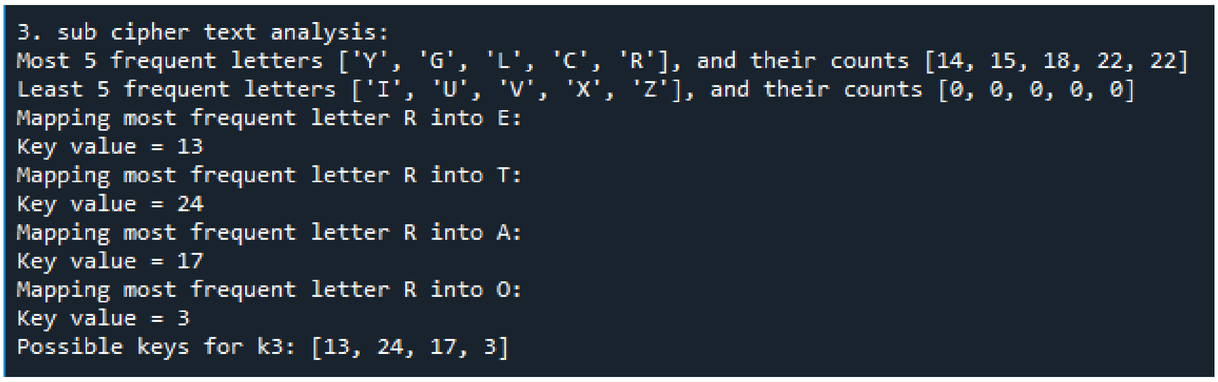

```
2. sub cipher text analysis:
Most 5 frequent letters ['O', 'N', 'T', 'A', 'E'], and their counts [14, 16, 16, 18, 20]
Least 5 frequent letters ['J', 'V', 'X', 'Z', 'F'], and their counts [0, 0, 0, 0, 1]
Mapping most frequent letter E into E:
Key value = 0
Mapping most frequent letter E into T:
Key value = 11
Mapping most frequent letter E into A:
Key value = 4
Mapping most frequent letter E into O:
Key value = 16
Possible keys for k2: [0, 11, 4, 16]
```



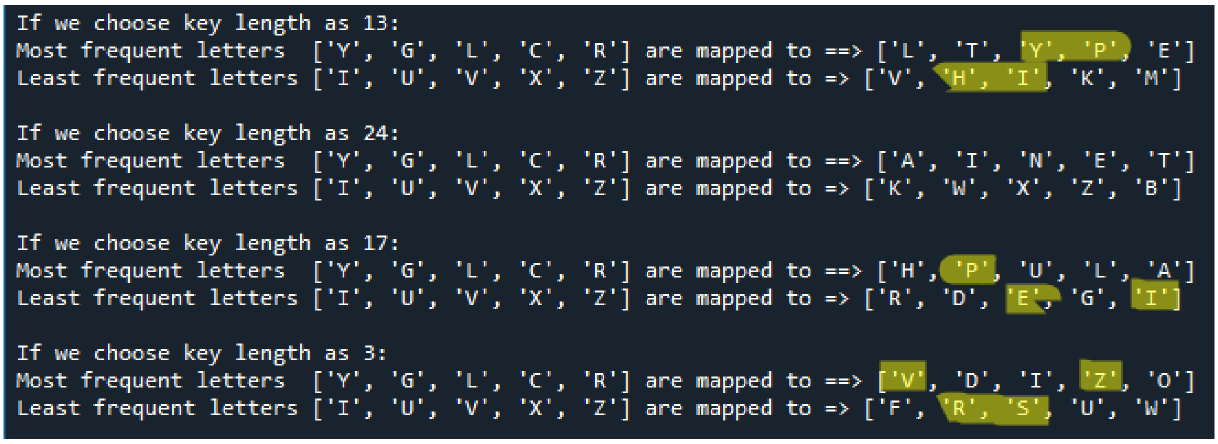
```
If we choose key length as 0:
Most frequent letters  ['O', 'N', 'T', 'A', 'E'] are mapped to ==> ['O', 'N', 'T', 'A', 'E']
Least frequent letters ['J', 'V', 'X', 'Z', 'F'] are mapped to => ['J', 'V', 'X', 'Z', 'F']

If we choose key length as 11:
Most frequent letters  ['O', 'N', 'T', 'A', 'E'] are mapped to ==> ['D', 'C', 'I', 'P', 'T']
Least frequent letters ['J', 'V', 'X', 'Z', 'F'] are mapped to => ['Y', 'K', 'M', 'O', 'U']

If we choose key length as 4:
Most frequent letters  ['O', 'N', 'T', 'A', 'E'] are mapped to ==> ['K', 'J', 'P', 'W', 'A']
Least frequent letters ['J', 'V', 'X', 'Z', 'F'] are mapped to => ['F', 'R', 'T', 'V', 'B']

If we choose key length as 16:
Most frequent letters  ['O', 'N', 'T', 'A', 'E'] are mapped to ==> ['Y', 'X', 'D', 'K', 'O']
Least frequent letters ['J', 'V', 'X', 'Z', 'F'] are mapped to => ['T', 'F', 'H', 'J', 'P']
```

Most promising one is **k2 = 0** as I explained before. I highlighted obvious inconsistencies.

Analysing 3. Sub Ciphertext:



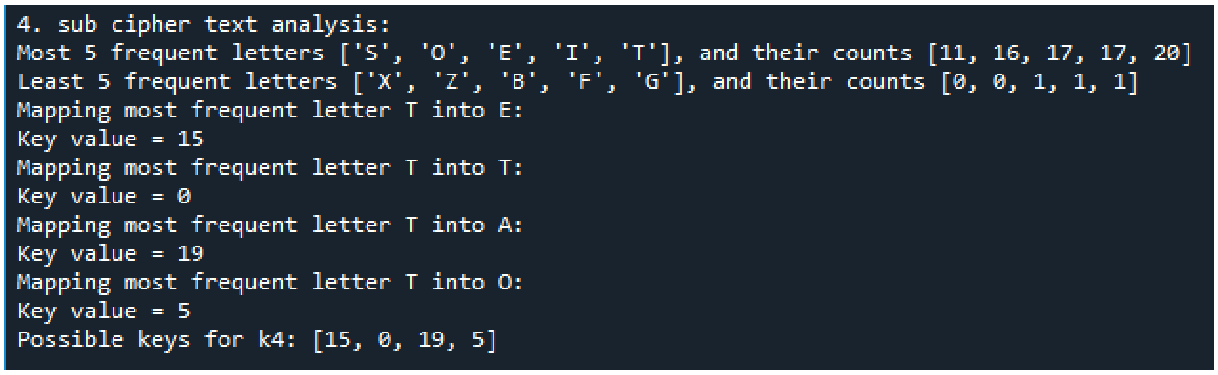
```
3. sub cipher text analysis:
Most 5 frequent letters ['Y', 'G', 'L', 'C', 'R'], and their counts [14, 15, 18, 22, 22]
Least 5 frequent letters ['I', 'U', 'V', 'X', 'Z'], and their counts [0, 0, 0, 0, 0]
Mapping most frequent letter R into E:
Key value = 13
Mapping most frequent letter R into T:
Key value = 24
Mapping most frequent letter R into A:
Key value = 17
Mapping most frequent letter R into O:
Key value = 3
Possible keys for k3: [13, 24, 17, 3]
```



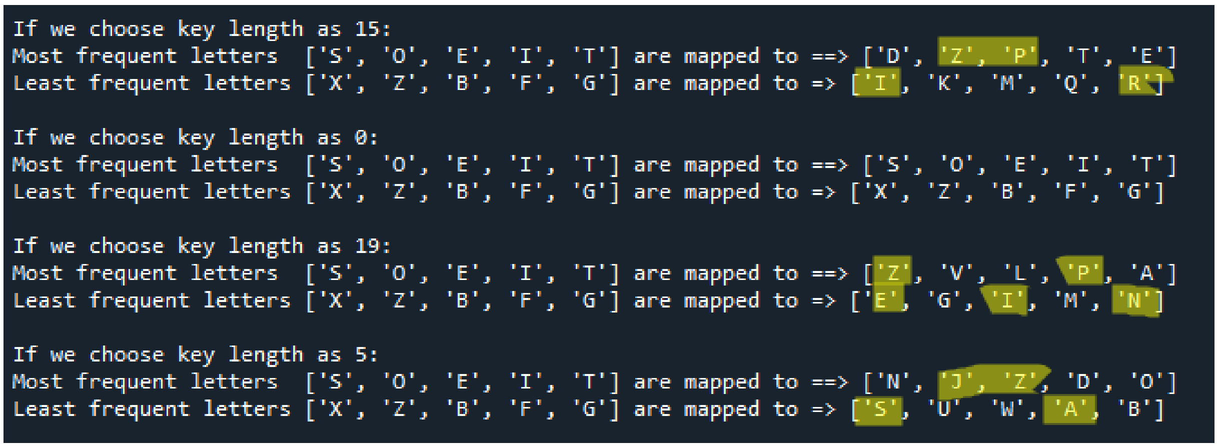
```
If we choose key length as 13:
Most frequent letters  ['Y', 'G', 'L', 'C', 'R'] are mapped to ==> ['L', 'T', 'Y', 'P', 'E']
Least frequent letters ['I', 'U', 'V', 'X', 'Z'] are mapped to => ['V', 'H', 'I', 'K', 'M']

If we choose key length as 24:
Most frequent letters  ['Y', 'G', 'L', 'C', 'R'] are mapped to ==> ['A', 'I', 'N', 'E', 'T']
Least frequent letters ['I', 'U', 'V', 'X', 'Z'] are mapped to => ['K', 'W', 'X', 'Z', 'B']

If we choose key length as 17:
Most frequent letters  ['Y', 'G', 'L', 'C', 'R'] are mapped to ==> ['H', 'P', 'U', 'L', 'A']
Least frequent letters ['I', 'U', 'V', 'X', 'Z'] are mapped to => ['R', 'D', 'E', 'G', 'I']

If we choose key length as 3:
Most frequent letters  ['Y', 'G', 'L', 'C', 'R'] are mapped to ==> ['V', 'D', 'I', 'Z', 'O']
Least frequent letters ['I', 'U', 'V', 'X', 'Z'] are mapped to => ['F', 'R', 'S', 'U', 'W']
```

Most promising one is **k3 = 24** as I explained before. I highlighted obvious inconsistencies.

Analysing 4. Sub Ciphertext:



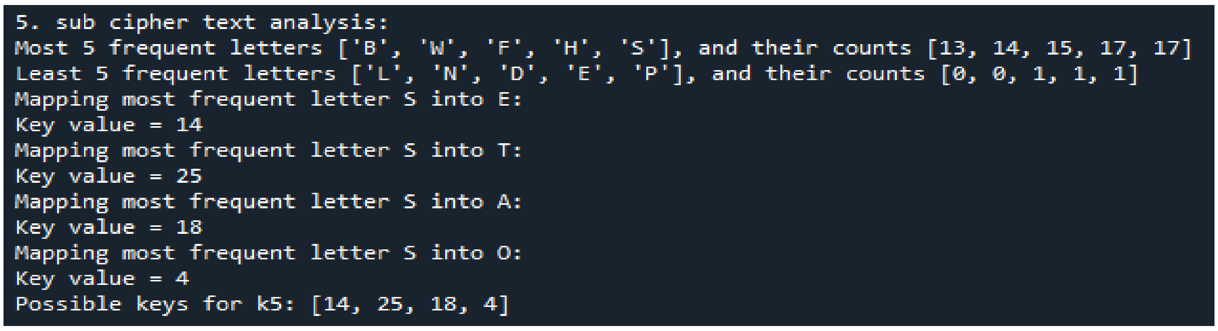
```
4. sub cipher text analysis:
Most 5 frequent letters ['S', 'O', 'E', 'I', 'T'], and their counts [11, 16, 17, 17, 20]
Least 5 frequent letters ['X', 'Z', 'B', 'F', 'G'], and their counts [0, 0, 1, 1, 1]
Mapping most frequent letter T into E:
Key value = 15
Mapping most frequent letter T into T:
Key value = 0
Mapping most frequent letter T into A:
Key value = 19
Mapping most frequent letter T into O:
Key value = 5
Possible keys for k4: [15, 0, 19, 5]
```



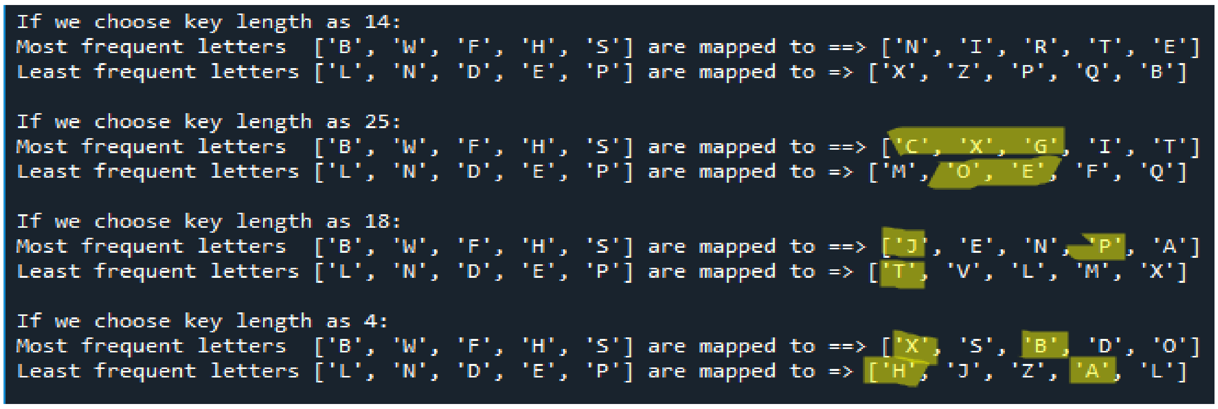
```
If we choose key length as 15:
Most frequent letters  ['S', 'O', 'E', 'I', 'T'] are mapped to ==> ['D', 'Z', 'P', 'T', 'E']
Least frequent letters ['X', 'Z', 'B', 'F', 'G'] are mapped to => ['I', 'K', 'M', 'Q', 'R']

If we choose key length as 0:
Most frequent letters  ['S', 'O', 'E', 'I', 'T'] are mapped to ==> ['S', 'O', 'E', 'I', 'T']
Least frequent letters ['X', 'Z', 'B', 'F', 'G'] are mapped to => ['X', 'Z', 'B', 'F', 'G']

If we choose key length as 19:
Most frequent letters  ['S', 'O', 'E', 'I', 'T'] are mapped to ==> ['Z', 'V', 'L', 'P', 'A']
Least frequent letters ['X', 'Z', 'B', 'F', 'G'] are mapped to => ['E', 'G', 'I', 'M', 'N']

If we choose key length as 5:
Most frequent letters  ['S', 'O', 'E', 'I', 'T'] are mapped to ==> ['N', 'J', 'Z', 'D', 'O']
Least frequent letters ['X', 'Z', 'B', 'F', 'G'] are mapped to => ['S', 'U', 'W', 'A', 'B']
```

Most promising one is **k4 = 0** as I explained before. I highlighted obvious inconsistencies.

Analysing 5. Sub Ciphertext:



```
5. sub cipher text analysis:
Most 5 frequent letters ['B', 'W', 'F', 'H', 'S'], and their counts [13, 14, 15, 17, 17]
Least 5 frequent letters ['L', 'N', 'D', 'E', 'P'], and their counts [0, 0, 1, 1, 1]
Mapping most frequent letter S into E:
Key value = 14
Mapping most frequent letter S into T:
Key value = 25
Mapping most frequent letter S into A:
Key value = 18
Mapping most frequent letter S into O:
Key value = 4
Possible keys for k5: [14, 25, 18, 4]
```

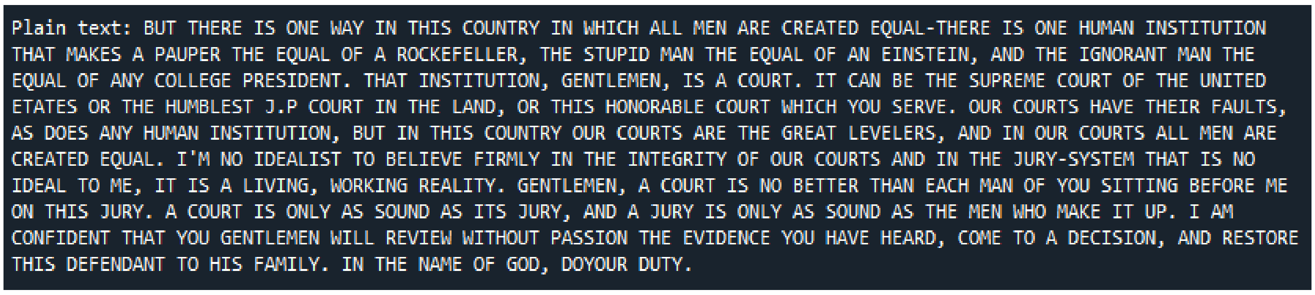


```
If we choose key length as 14:
Most frequent letters  ['B', 'W', 'F', 'H', 'S'] are mapped to ==> ['N', 'I', 'R', 'T', 'E']
Least frequent letters ['L', 'N', 'D', 'E', 'P'] are mapped to => ['X', 'Z', 'P', 'Q', 'B']

If we choose key length as 25:
Most frequent letters  ['B', 'W', 'F', 'H', 'S'] are mapped to ==> ['C', 'X', 'G', 'I', 'T']
Least frequent letters ['L', 'N', 'D', 'E', 'P'] are mapped to => ['M', 'O', 'E', 'F', 'Q']

If we choose key length as 18:
Most frequent letters  ['B', 'W', 'F', 'H', 'S'] are mapped to ==> ['J', 'E', 'N', 'P', 'A']
Least frequent letters ['L', 'N', 'D', 'E', 'P'] are mapped to => ['T', 'V', 'L', 'M', 'X']

If we choose key length as 4:
Most frequent letters  ['B', 'W', 'F', 'H', 'S'] are mapped to ==> ['X', 'S', 'B', 'D', 'O']
Least frequent letters ['L', 'N', 'D', 'E', 'P'] are mapped to => ['H', 'J', 'Z', 'A', 'L']
```

Most promising one is **k5 = 14** as I explained before. I highlighted obvious inconsistencies.

We found key as [7, 0, 24, 0, 14] which correspond to “HAYAO” as keyword.

Let’s try to decrypt our ciphertext with the key [7, 0, 24, 0, 14]:



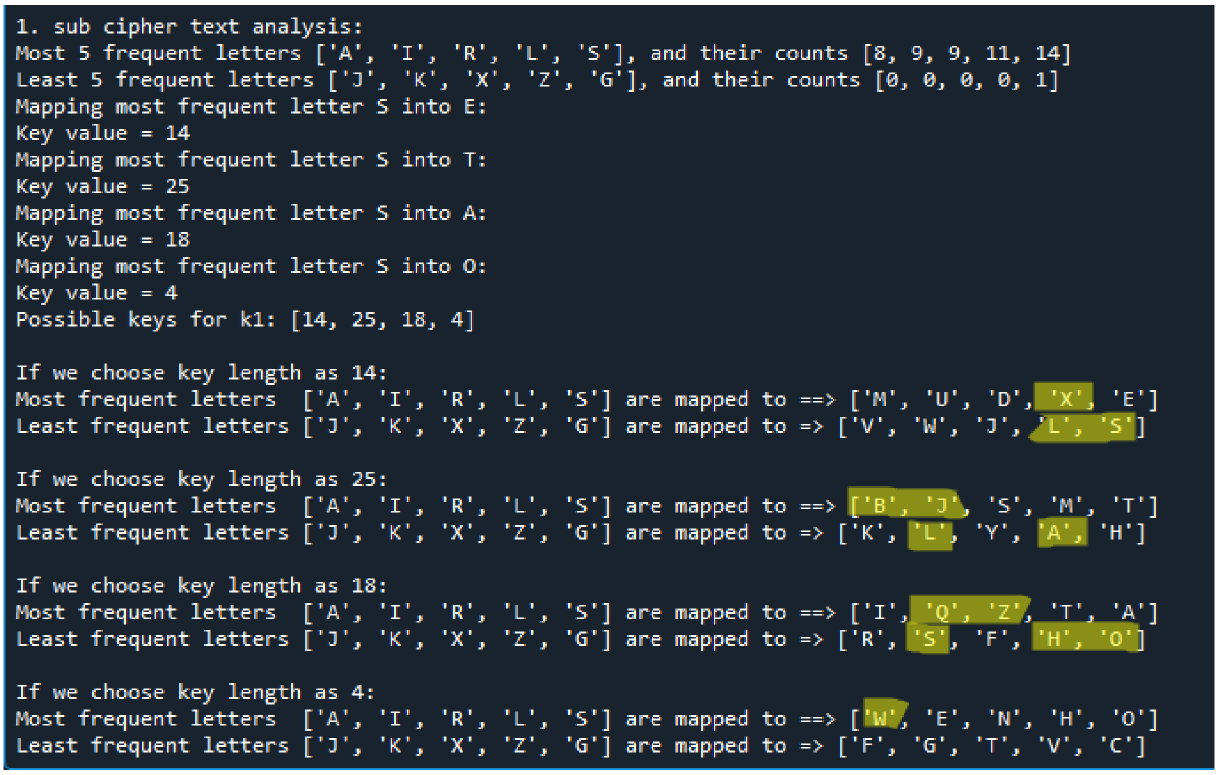
```
Plain text: BUT THERE IS ONE WAY IN THIS COUNTRY IN WHICH ALL MEN ARE CREATED EQUAL-THERE IS ONE HUMAN INSTITUTION
THAT MAKES A PAUPER THE EQUAL OF A ROCKEFELLER, THE STUPID MAN THE EQUAL OF AN EINSTEIN, AND THE IGNORANT MAN THE
EQUAL OF ANY COLLEGE PRESIDENT. THAT INSTITUTION, GENTLEMEN, IS A COURT. IT CAN BE THE SUPREME COURT OF THE UNITED
ETATES OR THE HUMBLEST J.P COURT IN THE LAND, OR THIS HONORABLE COURT WHICH YOU SERVE. OUR COURTS HAVE THEIR FAULTS,
AS DOES ANY HUMAN INSTITUTION, BUT IN THIS COUNTRY OUR COURTS ARE THE GREAT LEVELERS, AND IN OUR COURTS ALL MEN ARE
CREATED EQUAL. I'M NO IDEALIST TO BELIEVE FIRMLY IN THE INTEGRITY OF OUR COURTS AND IN THE JURY-SYSTEM THAT IS NO
IDEAL TO ME, IT IS A LIVING, WORKING REALITY. GENTLEMEN, A COURT IS NO BETTER THAN EACH MAN OF YOU SITTING BEFORE ME
ON THIS JURY. A COURT IS ONLY AS SOUND AS ITS JURY, AND A JURY IS ONLY AS SOUND AS THE MEN WHO MAKE IT UP. I AM
CONFIDENT THAT YOU GENTLEMEN WILL REVIEW WITHOUT PASSION THE EVIDENCE YOU HAVE HEARD, COME TO A DECISION, AND RESTORE
THIS DEFENDANT TO HIS FAMILY. IN THE NAME OF GOD, DOYOUR DUTY.
```

It looks okay for me.

It looks like we found the correct plain text. However, let’s try key length 7 as well.

Assuming key length 7

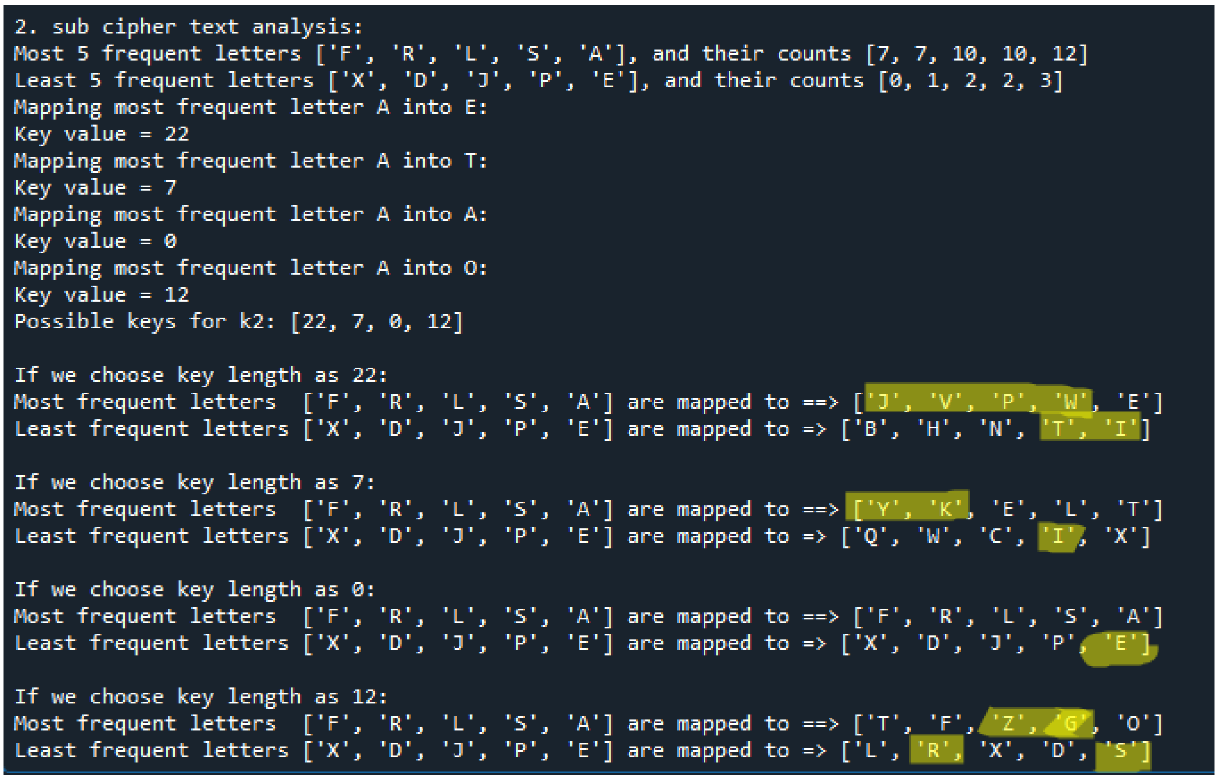
Analysing 1. Sub Ciphertext:



```
1. sub cipher text analysis:
Most 5 frequent letters ['A', 'I', 'R', 'L', 'S'], and their counts [8, 9, 9, 11, 14]
Least 5 frequent letters ['J', 'K', 'X', 'Z', 'G'], and their counts [0, 0, 0, 0, 1]
Mapping most frequent letter S into E:
Key value = 14
Mapping most frequent letter S into T:
Key value = 25
Mapping most frequent letter S into A:
Key value = 18
Mapping most frequent letter S into O:
Key value = 4
Possible keys for k1: [14, 25, 18, 4]

If we choose key length as 14:
Most frequent letters  ['A', 'I', 'R', 'L', 'S'] are mapped to ==> ['M', 'U', 'D', 'X', 'E']
Least frequent letters ['J', 'K', 'X', 'Z', 'G'] are mapped to => ['V', 'W', 'J', 'L', 'S']

If we choose key length as 25:
Most frequent letters  ['A', 'I', 'R', 'L', 'S'] are mapped to ==> ['B', 'J', 'S', 'M', 'T']
Least frequent letters ['J', 'K', 'X', 'Z', 'G'] are mapped to => ['K', 'L', 'Y', 'A', 'H']

If we choose key length as 18:
Most frequent letters  ['A', 'I', 'R', 'L', 'S'] are mapped to ==> ['I', 'Q', 'Z', 'T', 'A']
Least frequent letters ['J', 'K', 'X', 'Z', 'G'] are mapped to => ['R', 'S', 'F', 'H', 'O']

If we choose key length as 4:
Most frequent letters  ['A', 'I', 'R', 'L', 'S'] are mapped to ==> ['W', 'E', 'N', 'H', 'O']
Least frequent letters ['J', 'K', 'X', 'Z', 'G'] are mapped to => ['F', 'G', 'T', 'V', 'C']
```

Even though we have inconsistency let’s choose **k1 = 4 or 14.**

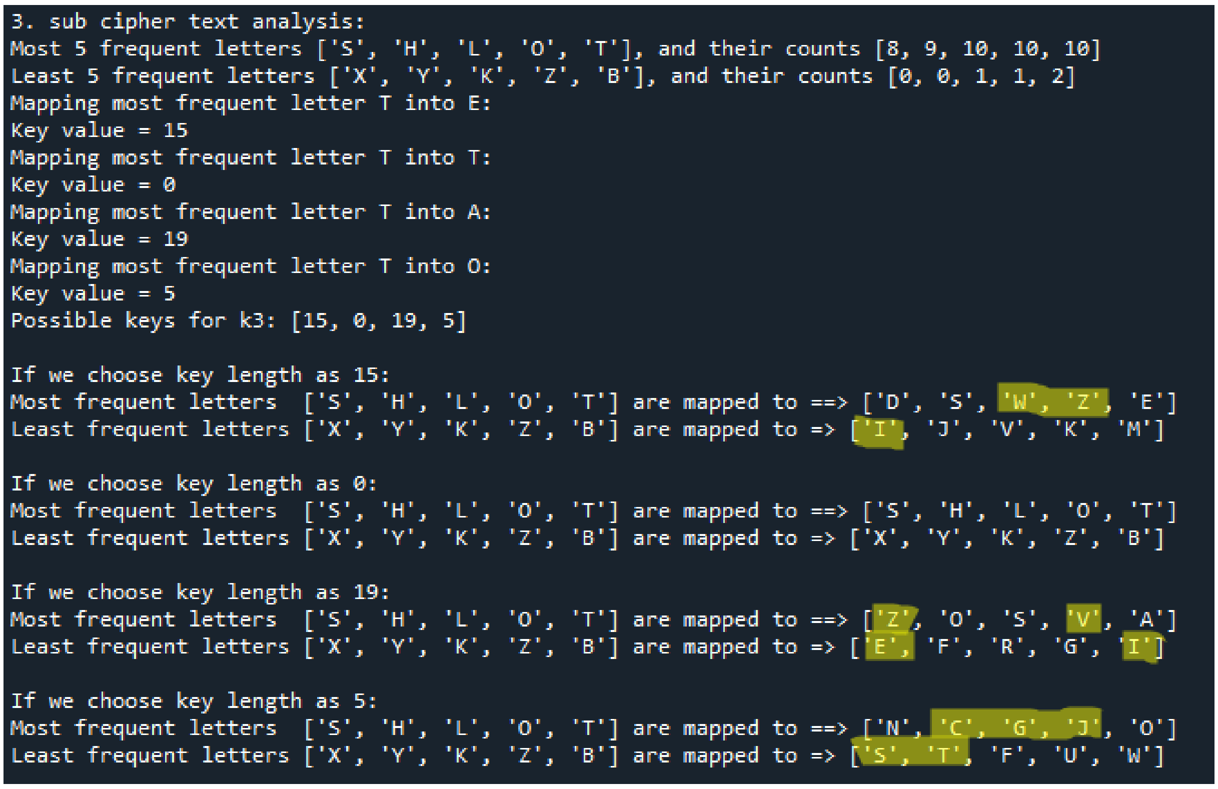
Analysing 2. Sub Ciphertext:



```
2. sub cipher text analysis:
Most 5 frequent letters ['F', 'R', 'L', 'S', 'A'], and their counts [7, 7, 10, 10, 12]
Least 5 frequent letters ['X', 'D', 'J', 'P', 'E'], and their counts [0, 1, 2, 2, 3]
Mapping most frequent letter A into E:
Key value = 22
Mapping most frequent letter A into T:
Key value = 7
Mapping most frequent letter A into A:
Key value = 0
Mapping most frequent letter A into O:
Key value = 12
Possible keys for k2: [22, 7, 0, 12]

If we choose key length as 22:
Most frequent letters  ['F', 'R', 'L', 'S', 'A'] are mapped to ==> ['J', 'V', 'P', 'W', 'E']
Least frequent letters ['X', 'D', 'J', 'P', 'E'] are mapped to => ['B', 'H', 'N', 'T', 'I']

If we choose key length as 7:
Most frequent letters  ['F', 'R', 'L', 'S', 'A'] are mapped to ==> ['Y', 'K', 'E', 'L', 'T']
Least frequent letters ['X', 'D', 'J', 'P', 'E'] are mapped to => ['Q', 'W', 'C', 'I', 'X']

If we choose key length as 0:
Most frequent letters  ['F', 'R', 'L', 'S', 'A'] are mapped to ==> ['F', 'R', 'L', 'S', 'A']
Least frequent letters ['X', 'D', 'J', 'P', 'E'] are mapped to => ['X', 'D', 'J', 'P', 'E']

If we choose key length as 12:
Most frequent letters  ['F', 'R', 'L', 'S', 'A'] are mapped to ==> ['T', 'F', 'Z', 'G', 'O']
Least frequent letters ['X', 'D', 'J', 'P', 'E'] are mapped to => ['L', 'R', 'X', 'D', 'S']
```

Even though we have inconsistency let’s choose **k2 = 0.**

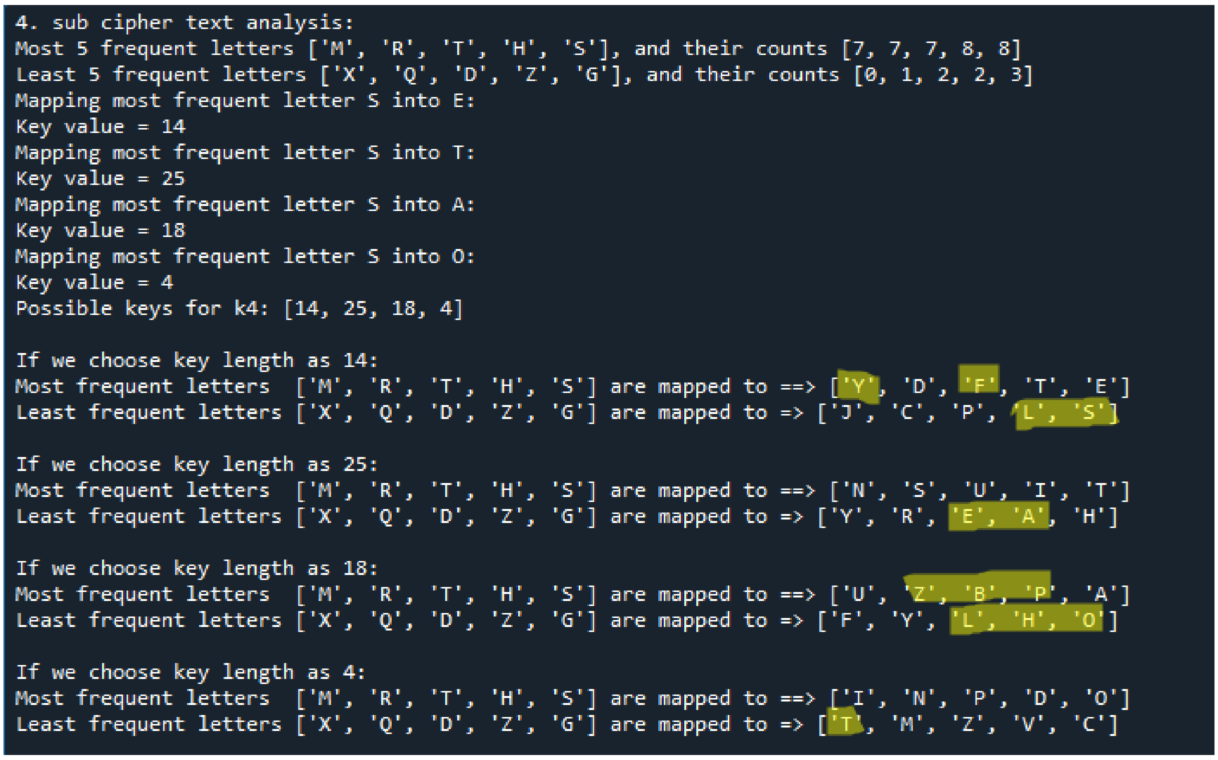
Analysing 3. Sub Ciphertext:



```
3. sub cipher text analysis:
Most 5 frequent letters ['S', 'H', 'L', 'O', 'T'], and their counts [8, 9, 10, 10, 10]
Least 5 frequent letters ['X', 'Y', 'K', 'Z', 'B'], and their counts [0, 0, 1, 1, 2]
Mapping most frequent letter T into E:
Key value = 15
Mapping most frequent letter T into T:
Key value = 0
Mapping most frequent letter T into A:
Key value = 19
Mapping most frequent letter T into O:
Key value = 5
Possible keys for k3: [15, 0, 19, 5]

If we choose key length as 15:
Most frequent letters  ['S', 'H', 'L', 'O', 'T'] are mapped to ==> ['D', 'S', 'W', 'Z', 'E']
Least frequent letters ['X', 'Y', 'K', 'Z', 'B'] are mapped to => ['I', 'J', 'V', 'K', 'M']

If we choose key length as 0:
Most frequent letters  ['S', 'H', 'L', 'O', 'T'] are mapped to ==> ['S', 'H', 'L', 'O', 'T']
Least frequent letters ['X', 'Y', 'K', 'Z', 'B'] are mapped to => ['X', 'Y', 'K', 'Z', 'B']

If we choose key length as 19:
Most frequent letters  ['S', 'H', 'L', 'O', 'T'] are mapped to ==> ['Z', 'O', 'S', 'V', 'A']
Least frequent letters ['X', 'Y', 'K', 'Z', 'B'] are mapped to => ['E', 'F', 'R', 'G', 'I']

If we choose key length as 5:
Most frequent letters  ['S', 'H', 'L', 'O', 'T'] are mapped to ==> ['N', 'C', 'G', 'J', 'O']
Least frequent letters ['X', 'Y', 'K', 'Z', 'B'] are mapped to => ['S', 'T', 'F', 'U', 'W']
```

Let’s choose **k3 = 0.**

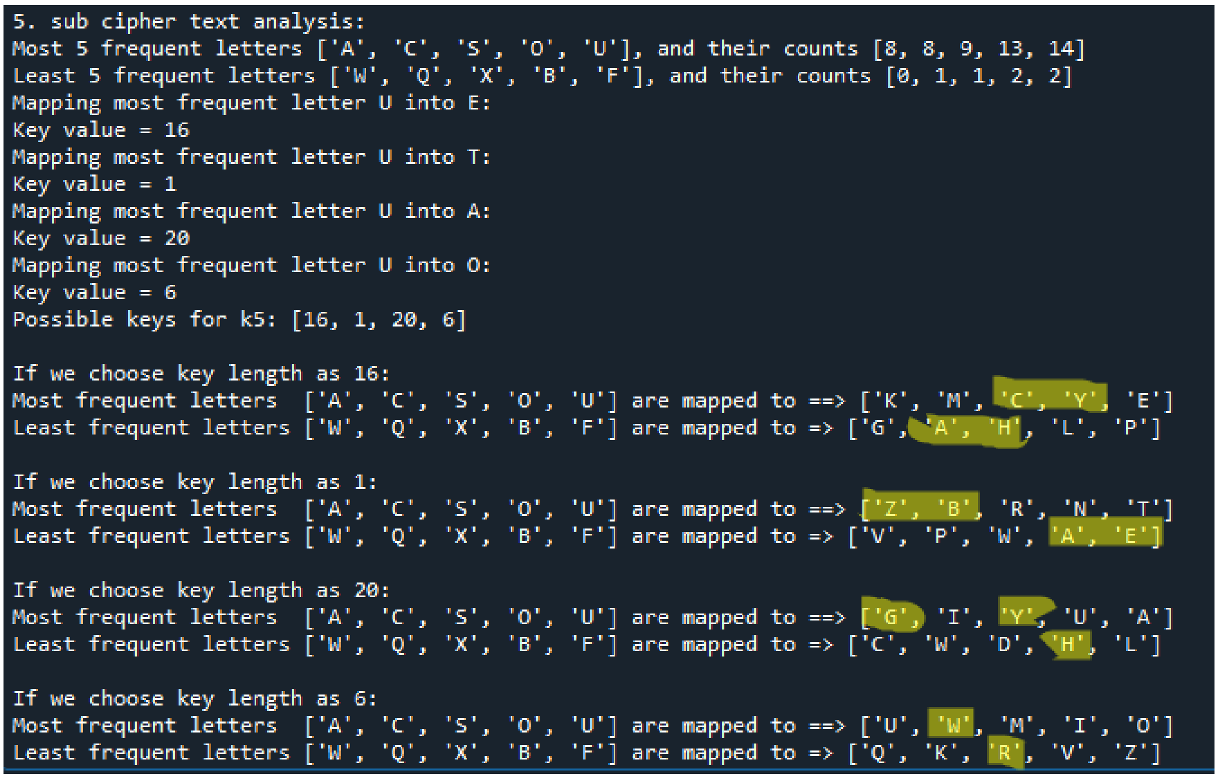
Analysing 4. Sub Ciphertext:



```
4. sub cipher text analysis:
Most 5 frequent letters ['M', 'R', 'T', 'H', 'S'], and their counts [7, 7, 7, 8, 8]
Least 5 frequent letters ['X', 'Q', 'D', 'Z', 'G'], and their counts [0, 1, 2, 2, 3]
Mapping most frequent letter S into E:
Key value = 14
Mapping most frequent letter S into T:
Key value = 25
Mapping most frequent letter S into A:
Key value = 18
Mapping most frequent letter S into O:
Key value = 4
Possible keys for k4: [14, 25, 18, 4]

If we choose key length as 14:
Most frequent letters  ['M', 'R', 'T', 'H', 'S'] are mapped to ==> ['Y', 'D', 'F', 'T', 'E']
Least frequent letters ['X', 'Q', 'D', 'Z', 'G'] are mapped to => ['J', 'C', 'P', 'L', 'S']

If we choose key length as 25:
Most frequent letters  ['M', 'R', 'T', 'H', 'S'] are mapped to ==> ['N', 'S', 'U', 'I', 'T']
Least frequent letters ['X', 'Q', 'D', 'Z', 'G'] are mapped to => ['Y', 'R', 'E', 'A', 'H']

If we choose key length as 18:
Most frequent letters  ['M', 'R', 'T', 'H', 'S'] are mapped to ==> ['U', 'Z', 'B', 'P', 'A']
Least frequent letters ['X', 'Q', 'D', 'Z', 'G'] are mapped to => ['F', 'Y', 'L', 'H', 'O']

If we choose key length as 4:
Most frequent letters  ['M', 'R', 'T', 'H', 'S'] are mapped to ==> ['I', 'N', 'P', 'D', 'O']
Least frequent letters ['X', 'Q', 'D', 'Z', 'G'] are mapped to => ['T', 'M', 'Z', 'V', 'C']
```

Even though we have inconsistency let’s choose **k4 = 4.**

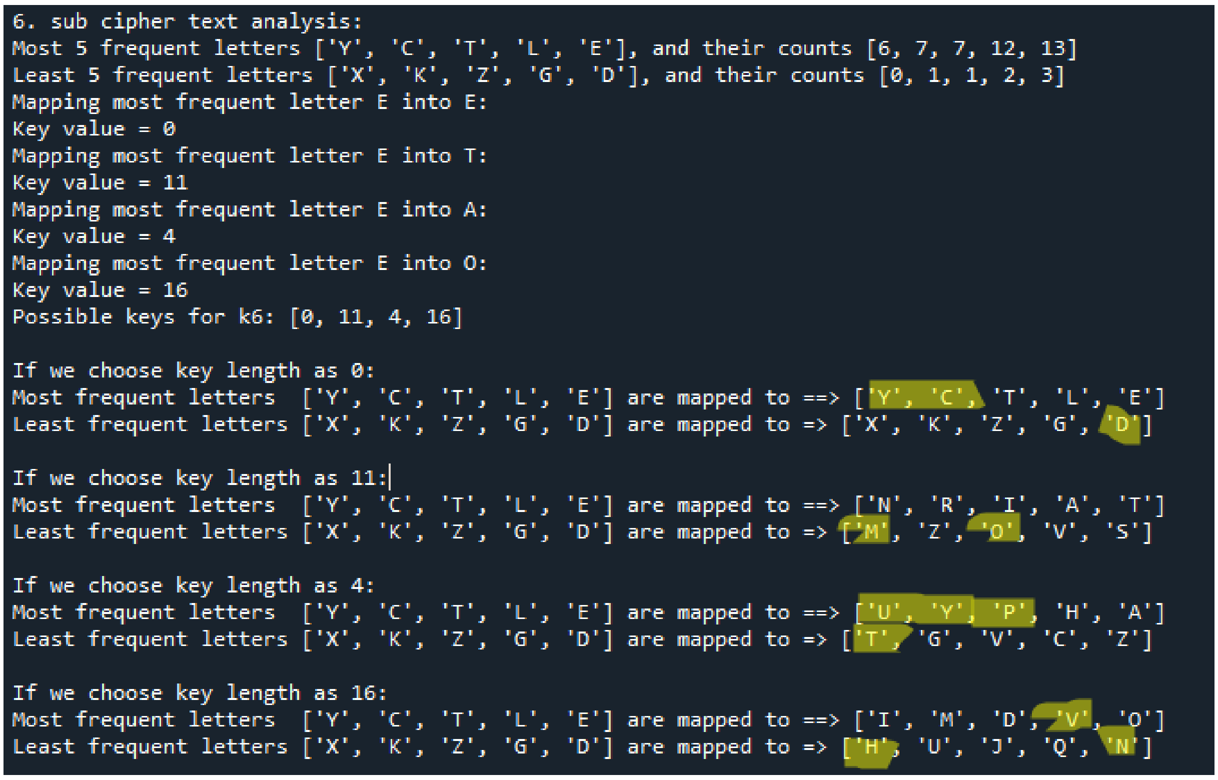
Analysing 5. Sub Ciphertext:



```
5. sub cipher text analysis:
Most 5 frequent letters ['A', 'C', 'S', 'O', 'U'], and their counts [8, 8, 9, 13, 14]
Least 5 frequent letters ['W', 'Q', 'X', 'B', 'F'], and their counts [0, 1, 1, 2, 2]
Mapping most frequent letter U into E:
Key value = 16
Mapping most frequent letter U into T:
Key value = 1
Mapping most frequent letter U into A:
Key value = 20
Mapping most frequent letter U into O:
Key value = 6
Possible keys for k5: [16, 1, 20, 6]

If we choose key length as 16:
Most frequent letters  ['A', 'C', 'S', 'O', 'U'] are mapped to ==> ['K', 'M', 'C', 'Y', 'E']
Least frequent letters ['W', 'Q', 'X', 'B', 'F'] are mapped to => ['G', 'A', 'H', 'L', 'P']

If we choose key length as 1:
Most frequent letters  ['A', 'C', 'S', 'O', 'U'] are mapped to ==> ['Z', 'B', 'R', 'N', 'T']
Least frequent letters ['W', 'Q', 'X', 'B', 'F'] are mapped to => ['V', 'P', 'W', 'A', 'E']

If we choose key length as 20:
Most frequent letters  ['A', 'C', 'S', 'O', 'U'] are mapped to ==> ['G', 'I', 'Y', 'U', 'A']
Least frequent letters ['W', 'Q', 'X', 'B', 'F'] are mapped to => ['C', 'W', 'D', 'H', 'L']

If we choose key length as 6:
Most frequent letters  ['A', 'C', 'S', 'O', 'U'] are mapped to ==> ['U', 'W', 'M', 'I', 'O']
Least frequent letters ['W', 'Q', 'X', 'B', 'F'] are mapped to => ['Q', 'K', 'R', 'V', 'Z']
```

Even though we have inconsistency let’s choose **k5 = 6.**

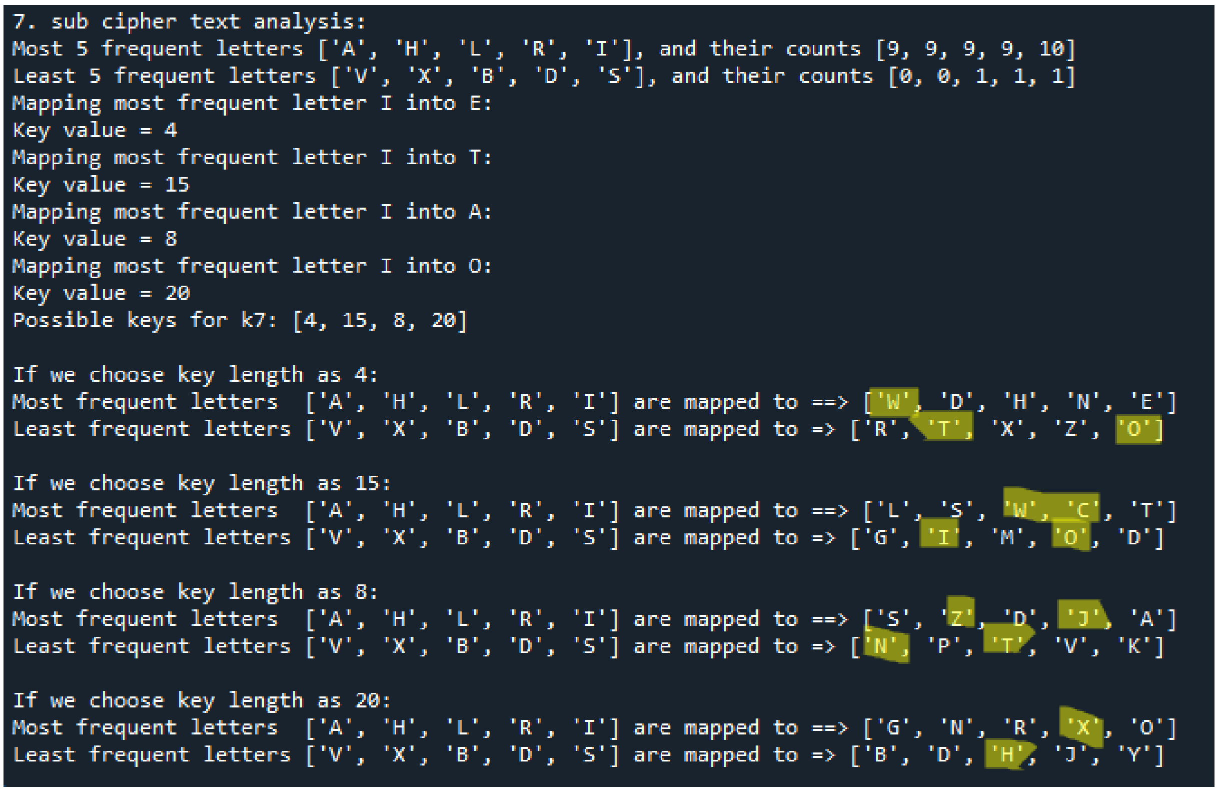
Analysing 6. Sub Ciphertext:



```
6. sub cipher text analysis:
Most 5 frequent letters ['Y', 'C', 'T', 'L', 'E'], and their counts [6, 7, 7, 12, 13]
Least 5 frequent letters ['X', 'K', 'Z', 'G', 'D'], and their counts [0, 1, 1, 2, 3]
Mapping most frequent letter E into E:
Key value = 0
Mapping most frequent letter E into T:
Key value = 11
Mapping most frequent letter E into A:
Key value = 4
Mapping most frequent letter E into O:
Key value = 16
Possible keys for k6: [0, 11, 4, 16]

If we choose key length as 0:
Most frequent letters  ['Y', 'C', 'T', 'L', 'E'] are mapped to ==> ['Y', 'C', 'T', 'L', 'E']
Least frequent letters ['X', 'K', 'Z', 'G', 'D'] are mapped to => ['X', 'K', 'Z', 'G', 'D']

If we choose key length as 11:
Most frequent letters  ['Y', 'C', 'T', 'L', 'E'] are mapped to ==> ['N', 'R', 'I', 'A', 'T']
Least frequent letters ['X', 'K', 'Z', 'G', 'D'] are mapped to => ['M', 'Z', 'O', 'V', 'S']

If we choose key length as 4:
Most frequent letters  ['Y', 'C', 'T', 'L', 'E'] are mapped to ==> ['U', 'Y', 'P', 'H', 'A']
Least frequent letters ['X', 'K', 'Z', 'G', 'D'] are mapped to => ['T', 'G', 'V', 'C', 'Z']

If we choose key length as 16:
Most frequent letters  ['Y', 'C', 'T', 'L', 'E'] are mapped to ==> ['I', 'M', 'D', 'V', 'O']
Least frequent letters ['X', 'K', 'Z', 'G', 'D'] are mapped to => ['H', 'U', 'J', 'Q', 'N']
```

Even though we have inconsistency let’s choose **k6 = 11.**

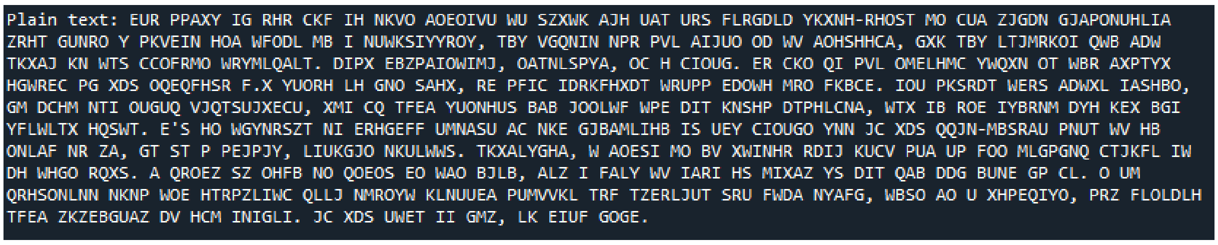
Analysing 7. Sub Ciphertext:



```
7. sub cipher text analysis:
Most 5 frequent letters ['A', 'H', 'L', 'R', 'I'], and their counts [9, 9, 9, 9, 10]
Least 5 frequent letters ['V', 'X', 'B', 'D', 'S'], and their counts [0, 0, 1, 1, 1]
Mapping most frequent letter I into E:
Key value = 4
Mapping most frequent letter I into T:
Key value = 15
Mapping most frequent letter I into A:
Key value = 8
Mapping most frequent letter I into O:
Key value = 20
Possible keys for k7: [4, 15, 8, 20]

If we choose key length as 4:
Most frequent letters  ['A', 'H', 'L', 'R', 'I'] are mapped to ==> ['W', 'D', 'H', 'N', 'E']
Least frequent letters ['V', 'X', 'B', 'D', 'S'] are mapped to => ['R', 'T', 'X', 'Z', 'O']

If we choose key length as 15:
Most frequent letters  ['A', 'H', 'L', 'R', 'I'] are mapped to ==> ['L', 'S', 'W', 'C', 'T']
Least frequent letters ['V', 'X', 'B', 'D', 'S'] are mapped to => ['G', 'I', 'M', 'O', 'D']

If we choose key length as 8:
Most frequent letters  ['A', 'H', 'L', 'R', 'I'] are mapped to ==> ['S', 'Z', 'D', 'J', 'A']
Least frequent letters ['V', 'X', 'B', 'D', 'S'] are mapped to => ['N', 'P', 'T', 'V', 'K']

If we choose key length as 20:
Most frequent letters  ['A', 'H', 'L', 'R', 'I'] are mapped to ==> ['G', 'N', 'R', 'X', 'O']
Least frequent letters ['V', 'X', 'B', 'D', 'S'] are mapped to => ['B', 'D', 'H', 'J', 'Y']
```

Even though we have inconsistency let’s choose **k7 = 20.**

Let’s try to decrypt our ciphertext with the key [4, 0, 0, 4, 6, 11, 20]:



```
Plain text: EUR PPAXY IG RHR CKF IH NKVO AOEOIVU WU SZXWK AJH UAT URS FLRGDLD YKXNH-RHOST MO CUA ZJGDN GJAPONUHLIA
ZRHT GUNRO Y PKVEIN HOA WFODL MB I NUWKSIYYROY, TBY VGQNIN NPR PVL AIJUO OD WV AOHSHHCA, GXK TBY LTJMRKOI QWB ADW
TKXAJ KN WTS CCOFRMO WRYMLQALT. DIPX EBZPAIOWIMJ, OATNLSPYA, OC H CIOUG. ER CKO QI PVL OMELHMC YWQXN OT WBR AXPTYX
HGWREC PG XDS OQEQFHSR F.X YUORH LH GNO SAHX, RE PFIC IDRKFHXDT WRUPP EDOWH MRO FKBCE. IOU PKSRDT WERS ADWXL IASHBO,
GM DCHM NTI OUGUQ VJQTSUJXECU, XMI CQ TFEA YUONHUS BAB JOOLWF WPE DIT KNSHP DTPHLCNA, WTX IB ROE IYBRNM DYH KEX BGI
YFLWLTX HQSWT. E'S HO WGYNRSZT NI ERHGEFF UMNASU AC NKE GJBAMLIHB IS UEY CIOUGO YNN JC XDS QQJN-MBSRAU PNUT WV HB
ONLAF NR ZA, GT ST P PEJPJY, LIUKGJO NKULWWS. TKXALYGHA, W AOESI MO BV XWINHR RDIJ KUCV PUA UP FOO MLGPGNQ CTJKFL IW
DH WHGO RQXS. A QROEZ SZ OHFB NO QOEOS EO WAO BJLB, ALZ I FALY WV IARI HS MIXAZ YS DIT QAB DDG BUNE GP CL. O UM
QRHSONLNN NKNP WOE HTRPZLIWC QLLJ NMROYW KLNUUEA PUMVVKL TRF TZERLJUT SRU FWDA NYAFG, WBSO AO U XHPEQIYO, PRZ FLOLDLH
TFEA ZKZEBGUAZ DV HCM INIGLI. JC XDS UWET II GMZ, LK EIUF GOGE.
```

It is not meaningful.

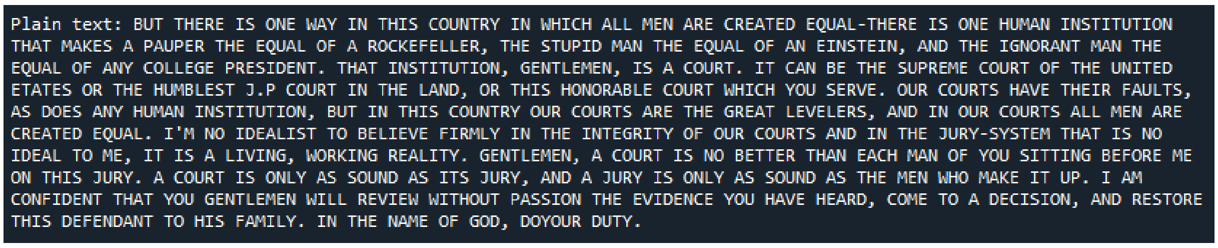
Consequently, for bonus question,

Our key length is 5.

Our key is [7, 0, 24, 0, 14]

Our keyword is “HAYAO”

Our plaintext is:



```
Plain text: BUT THERE IS ONE WAY IN THIS COUNTRY IN WHICH ALL MEN ARE CREATED EQUAL-THERE IS ONE HUMAN INSTITUTION
THAT MAKES A PAUPER THE EQUAL OF A ROCKEFELLER, THE STUPID MAN THE EQUAL OF AN EINSTEIN, AND THE IGNORANT MAN THE
EQUAL OF ANY COLLEGE PRESIDENT. THAT INSTITUTION, GENTLEMEN, IS A COURT. IT CAN BE THE SUPREME COURT OF THE UNITED
ETATES OR THE HUMBLEST J.P COURT IN THE LAND, OR THIS HONORABLE COURT WHICH YOU SERVE. OUR COURTS HAVE THEIR FAULTS,
AS DOES ANY HUMAN INSTITUTION, BUT IN THIS COUNTRY OUR COURTS ARE THE GREAT LEVELERS, AND IN OUR COURTS ALL MEN ARE
CREATED EQUAL. I'M NO IDEALIST TO BELIEVE FIRMLY IN THE INTEGRITY OF OUR COURTS AND IN THE JURY-SYSTEM THAT IS NO
IDEAL TO ME, IT IS A LIVING, WORKING REALITY. GENTLEMEN, A COURT IS NO BETTER THAN EACH MAN OF YOU SITTING BEFORE ME
ON THIS JURY. A COURT IS ONLY AS SOUND AS ITS JURY, AND A JURY IS ONLY AS SOUND AS THE MEN WHO MAKE IT UP. I AM
CONFIDENT THAT YOU GENTLEMEN WILL REVIEW WITHOUT PASSION THE EVIDENCE YOU HAVE HEARD, COME TO A DECISION, AND RESTORE
THIS DEFENDANT TO HIS FAMILY. IN THE NAME OF GOD, DOYOUR DUTY.
```

Corresponding codes can be found in the CS411_HW1_bonus.py