Data Sec HW 4

Problem 1. Find the last digit of 9^{123456} .
$9^1 = 9$
$9^2 = 81$
$9^3 = 729$
$9^4 = 6561$
...
You can see that there is a pattern. If the exponent is odd, then the last digit will be a 9.
If the exponent is even, the last digit will be 1. In this example, since our exponent is 123456, which is an even number, we know that the last digit is a 1.
We can also say that 123456 % 2 = 0. Then using the remainder, 0, we say $9^0 = 1$
If we had an odd exponent, say 123457, we can still do the same operations. 123457 % 2 = 1, $9^1 = 9$ as our remainder.

Problem 2. What is (1! + 2! + 3! + 4! + 5! + 6! + · · ·) mod 12?
$\sum_{i=1}^{n} i$! mod 12.
1 + 2 + 6 + 24 + 120 + 720 + 5040 + ... mod 12
Can also look at it as:
(1) + (1 2) + (1 2 3) + (1 2 3 4) + (1 2 3 4 5) + ... mod 12
This can be (1 mod 12) + (2 mod 12) + (6 mod 12) + (24 mod 12) + (120 mod 12) + (720 mod 12) + (5040 mod 12) + (40320 mod 12)
1 + 2 + 6 + 0 + 0 + 0 + 0 + 0 + ... (infinite # of zeroes)
That means the remainder for this infinite sum of factorials will be 1 + 2 + 6 = 9.

Problem 3. In class, we looked at the definition of perfect secrecy, which states that the ciphertext C doesn't carry any information about M and hence it doesn't increase the adversary's probability of guessing C correctly.

$$P[M = m|C = c] = P[M = m]$$

(a) Show that the above definition is equivalent to the following definition.

$$P[M = m, C = c] = P[M = m] \cdot P[C = c]$$

```
$P[M = m | C = c] = P[M = m]$
$P[M = m] = \frac{P[M = m,C = c]}{P[C = c]}$  By the definition of conditional probability
$P[M = m] * P[C = c] = P[M = m, C = c]$ divide both sides by $P[C = c]$
$P[M = m, C = c] = P[M = m] * P[C = c]$ flip the equation
```

(b) Consider a simple one-time pad for binary string of length 2 in which the message space, ciphertext space, and key space are defined as $M = C = K = 00, 01, 10, 11$. It is known that the one-time pad is perfectly secret if the key k is chosen uniformly random, i.e., P[k = 00] = $P[k = 01] = P[10] = P[11] = \frac{1}{4}$. This can be proved by showing that, for any m_1, m_2, c, the method satisfies

$P[Enc(k_1, m_1) = c] = P[Enc(k_2, m_2) = c]$.(1)

Let's look at the first probability. Given m1 and c,

$P[Enc(k, m_1) = c] = P[k \oplus m_1 = c] = P[k = c \oplus m_1] = \frac{1}{4}$.

The above means that m_1 will be encrypted into c if the specific key k = c ⊕ m_1 is chosen. Since in one-time pad all keys are equally likely, m_1 has the probability of $\frac{1}{4}$ to be encrypted into c. Observe that, with the same reasoning, m2 also has the same probability to be encrypted into c, i.e., $P[Enc(k, m_2) = c] = \frac{1}{4}$.

Suppose that you decided to remove the key k = 00 from the key space as it sends the message in clear. When k = 00, we have

$c = Enc(k, m) = 00 \oplus m = m.$

Prove or disprove whether the modified method is still perfectly secret. In other words, either show that (1) still holds true with k = 00 removed or provide a counter example (m_1, m_2, c) for which

$P[Enc(k_1, m_1) = c]! = P[Enc(k_2, m_2) = c]$.

Hint: given fixed m and c, there exists a unique k such that m ⊕ k = c.
If you do the exclusive or of 0 with any bit string, for example
000000000 ⊕ 100011001, you get the same number: 100011001. This is one of the properties of exclusive or. In practice, you would not use this key because the plaintext would be the same as the ciphertext. By removing this key value, this would no longer satisfy perfect privacy.
A counter example would be if $m_1 = 00$, $m_2 = 11$ and $c = 11$
$k \oplus m_1 = 11$, which means $k = 11$
$k \oplus m_2 = 11$, which means $k = 00$
Since there are 3 keys in the keyspace {01, 10, 11}, the probability of choosing the first example (k = 11) is $\frac{1}{3}$ since the probability distribution is it's uniformly distributed. However, in our second example, k = 00 is not in our keyspace, so the probability of choosing k = 00 is 0.
$P[Enc(k, m_1) = c]! = P[Enc(k, m_2) = c]$
$\frac{1}{3}! = 0$

Problem 4. Consider the following cryptosystem in which
• the message space (or plaintext space) is M = {a, b, c},
• the keyspace is K = $\{k_1, k_2, k_3\}$,and
• the ciphertext space is C = {1, 2, 3}.

The following table describes how the encryption is done, i.e., Enc(k, m).

K/M	a	b	c
k_1	2	3	1
k_2	3	1	2
k_3	1	2	3

For example, Enc(k_1,b) = 3. This means the plaintext 'b' is mapped to the ciphertext 3 if the key k2 is chosen. Suppose that messages occur with probability

• $P(M = a) = \frac{1}{4}$,
• $P(M = b) = \frac{1}{2}$, and
• $P(M = c) = \frac{1}{4}$.

Let's further assume the probability distribution over keys is given by
• $P(K = k_1) = \frac{1}{2}$,
• $P(K = k_2) = \frac{1}{4}$, and
• $P(K = k_3) = \frac{1}{4}$.

(a) Compute the probability of each ciphertext, $P(C = 1), P(C = 2)$, and $P(C = 3)$.
$P[C = 1] = \sum_{m \in M} P[C = 1|M = m] * P[M = m]$
=
$P[K = k_1] * P[M = c] = \frac{1}{2} * \frac{1}{4} = \frac{1}{8}$
+
$P[K = k_2] * P[M = b] = \frac{1}{4} * \frac{1}{2} = \frac{1}{8}$
+
$P[K = k_3] * P[M = a] = \frac{1}{4} * \frac{1}{4} = \frac{1}{16}$
= $\frac{1}{8} + \frac{1}{8} + \frac{1}{16}$ = $\frac{5}{16}$
$P[C = 2]$
=
$P[K = k_1] * P[M = a] = \frac{1}{2} * \frac{1}{4} = \frac{1}{8}$
$P[K = k_2] * P[M = c] = \frac{1}{4} * \frac{1}{4} = \frac{1}{16}$
$P[K = k_3] * P[M = b] = \frac{1}{4} * \frac{1}{2} = \frac{1}{8}$
$= \frac{1}{8} + \frac{1}{8} + \frac{1}{16} = \frac{5}{16}$

$P[C = 3]$
=
$P[K = k_1] * P[M = b] = \frac{1}{2} * \frac{1}{2} = \frac{1}{4}$
$P[K = k_2] * P[M = a] = \frac{1}{4} * \frac{1}{4} = \frac{1}{16}$
$P[K = k_3] * P[M = c = \frac{1}{4} * \frac{1}{4} = \frac{1}{16}$
$= \frac{1}{4} + \frac{1}{16} + \frac{1}{16}$ = $\frac{6}{16} = \frac{3}{8}$

(b) Compute $P[M = a|C = 1]$.
$P[M = a|C = 1] = \frac{P[C=1|M=a]*P[M=a]}{P[C=1|M=a]*P[M=a]+P[C=1|M=b]*P[M=b]+P[C=1|M=c]*P[M=c]}$
= $\frac{P[K=k_3]*P[M=a]}{P[C=1]}$
= $\frac{\frac{1}{4}*\frac{1}{4}}{\frac{5}{16}}$ = $\frac{\frac{1}{16}}{\frac{5}{16}}$ = $\frac{1}{16} * \frac{16}{5} = \frac{16}{80}$ = $\frac{1}{5}$
$P[C = 1]$ is already calculated above in (a)

Problem 5. In class, we learned a simple symmetric shift cipher (a.k.a. Caesar cipher). The secret key K is an integer in {0,1,...,25}. As always, we map each alphabet letter x ∊ {A, B, ...,Z} to an integer {0,1,...,25}. The encryption and description are defined by

$Enc(x, k) = (x + k)$ mod 26, $Dec(x, k) = (x - k)$ mod 26.

1. (a) As a warm-up, encrypt “CSCI IS COOL” using a Caesar cipher with k=’F’.
   F = 5

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

```
The encrypted message would be: "HXHN NX HTTQ" using the shifted Caesar's cipher mapping above.
```

1. (b) Here’s the ciphertext generated by the shift cipher. Find the corresponding plaintext (please explain the approach/strategy you used to find the key).

   IWXHFJTHIXDCXHTPHN
   The approach to use in this case is brute force since there are only 26 possible keys in the keyspace. I implemented a quick program to decipher this.

```python
def caesar_decrypt(ciphertext, shift):
    result = ""
    for char in ciphertext:
        if char.isalpha():
            base = ord('A')
            result += chr((ord(char.upper()) - base - shift) % 26 + base)
        else:
            result += char
    return result

ciphertext = "IWXHFJTHIXDCXHTPHN"

for k in range(1, 26):
    decrypted = caesar_decrypt(ciphertext, k)
    print(f"k = {k}, {decrypted}")
```

   The output is as shown (notice k = 15):

```
k = 1, HVWGEISGHWCBWGSOGM
k = 2, GUVFDHRFGVBAVFRNFL
k = 3, FTUECGQEFUAZUEQMEK
k = 4, ESTDBFPDETZYTDPLDJ
k = 5, DRSCAEOCDSYXSCOKCI
k = 6, CQRBZDNBCRXWRBNJBH
k = 7, BPQAYCMABQWVQAMIAG
k = 8, AOPZXBLZAPVUPZLHZF
k = 9, ZNOYWAKYZOUTOYKGYE
k = 10, YMNXVZJXYNTSNXJFXD
k = 11, XLMWUYIWXMSRMWIEWC
k = 12, WKLVTXHVWLRQLVHDVB
k = 13, VJKUSWGUVKQPKUGCUA
k = 14, UIJTRVFTUJPOJTFBTZ
k = 15, THISQUESTIONISEASY
k = 16, SGHRPTDRSHNMHRDZRX
k = 17, RFGQOSCQRGMLGQCYQW
k = 18, QEFPNRBPQFLKFPBXPV
k = 19, PDEOMQAOPEKJEOAWOU
k = 20, OCDNLPZNODJIDNZVNT
k = 21, NBCMKOYMNCIHCMYUMS
k = 22, MABLJNXLMBHGBLXTLR
k = 23, LZAKIMWKLAGFAKWSKQ
k = 24, KYZJHLVJKZFEZJVRJP
k = 25, JXYIGKUIJYEDYIUQIO
```

The table was taking too long. The Decrypted Ciphertext reads: THISQUESTIONISEASY

Keys	Plaintext
k = 1	JXYIGKUIJYEDYIUQIO
k = 2	KYZJHLVJKZFEZJVRJP
k = 3	
k = 4	
k = 5	
k = 6	

Problem 6. In class, we learned that one-time pad is not secure if the same key is repeatedly used for encrypting messages. Suppose that your classmate Alice is encrypting two messages, denoted by m1 and m2, each of which is an n-bit string, using the one-time pad. She knew that it's not safe to re-use the key k, so she decided to encrypt as follows:

• $c_1 = m_1 \oplus$ k and
• $c_2 = m_2 \oplus$ k′,
where k′ = 0 ⊕ 1 ⊕ k.

Is it safe to release c1 and c2? Explain your answer with justification.
If it's not safe, explain what information (about m1 and m2) can be leaked from c1 and c2.
If you plug in k' = 0 ⊕ 1 ⊕ k, you get
$c_2 = m_2 \oplus$ (0 ⊕ 1 ⊕ k)
If you try to XOR c_1 and c_2, you get
$c_1 \oplus c_2$
or
($m_1 \oplus$ k) ⊕ ($m_2 \oplus$ (0 ⊕ 1 ⊕ k))
You can get rid of the parenthesis because you're doing exclusive OR.

$m_1 \oplus k \oplus m_2 \oplus 0 \oplus 1 \oplus k$
reorder them because communicative:
$k \oplus k \oplus m_1 \oplus m_2 \oplus 0 \oplus 1$
Anything XOR by itself is 0.
$0 \oplus 0 \oplus m_1 \oplus m_2 \oplus 1$
$m_1 \oplus m_2 \oplus 1$
While this does not give you the key, even though you don't know the key, you can still establish a relationship between m_1 and m_2. In this case, it can tell you whether the corresponding bit in m_1 and m_2 are the same, or different. Because if $m_1[i] \oplus m_2[i]$ results in 0, then we know that the bits are the same. If they result in 1, we know that the bits are different. The adv can't see the original messages, but they can learn a pattern. Alice should not have had a deterministic relationship between k and k'.

Problem 7. Suppose that Alice and Bob would like to share an n-bit key for their encrypted communications over the internet. Let k_a and k_b denote the keys held by Alice and Bob, respectively. Alice wants to test if her key k_a is indeed the same with the key k_b Bob has. She came up with the following protocol to verify if $k_a = k_b$.

1. Alice generates a random binary string m = $b_1 b_2 \dots b_n$, where $b_i \in \{0, 1\}$.
2. Alice sends $a = k_a \oplus$ m to Bob.
3. Bob receives a, computes b = $k_b \oplus$ a, and sends it to Alice.
4. Alice concludes $k_a = k_b$ (i.e., Bob indeed received the key she sent) if m = b and $k_a \neq k_b$ otherwise.

Suppose that there exists an adversary on the internet who can eavesdrop (but cannot modify) the messages Alice and Bob exchange. Show how the adversary exactly can figure out the key they shared (i.e., k_a or k_b).
If the adversary can eavesdrop or do some packet sniffing for the encrypted message that Alice sends to Bob, the adv can intercept the encrypted message a, and the encrypted message b.
a $\oplus$ b
a $\oplus$ (k_b $\oplus$ a)
drop parenthesis
a $\oplus$ a $\oplus$ k_b
anything XOR itself is 0
0 $\oplus$ k_b
anything XOR 0 is itself
k_b
By taking the XOR of a and b, you're able to cancel out the encrypted message that Alice sends and derive the key that was used in decrypting the message that Alice sends.

Problem 13. Using the tools you wrote, decrypt the message. Please explain important milestones in your trial-and-error process and rationale behind a specific guess.

- Each milestone should show how you arrived at a specific guess and the deciphered text resulted from the guess.

Before I start, the ciphertext reads:
gsrhrhgsvkozrmgvcggszgdzhvmxibkgvwfhrmtgsvhfyhgrgfgrlmzmwxlmtizgfozgrlmhblfszevhfxxvhhufoobwvxrksvivwgsvnvhhztvz
mwryvorvevgszgblfmldszevyvggvifmwvigzmwrmtlmsldgsvyzhrxhfyhgrgfgrlmxrksvidliphgsvgirzozmwviilikilxvhhdzhylirmtyfglmxv
zuvdovggvihuzoormglgsvkozxvhgsvivnzrmrmtlmvhdroowlhljfrxpobuilnsvivrznvmgvirmtgsvnvzmrmtovhhhvmgvmxvhglnzpvgsvw
vxrksvivzhrvi

If I take the unigram of the ciphertext and sort from most commonly found letter to least commonly found letter, I get v = 50 as my most common letter. I can try to map this to e because e is the most commonly found letter in english alphabet.

This results in :

```
________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___e_
```

Next I can count the second most frequently found letter in the ciphertext, which results in g = 36.
According to the lecture slides, t is the second most commonly used letter in the English alphabet. After replacing all g's with t's in the ciphertext, you get:

```
t_____t_e_____te_tt__t___e_____te______t_e____t_t_t____________t___t__________e____e________e____e_e_t
_e_e____e_____e__e_et__t_________e_ette____e_t___________t_e_________t_t_t_______e______t_et_______e__
______e_____________t___e__e__ette________t_t_e____e_t_e_e_________e_____________________e_e___e_te___
_t_e_e______e___e_te__e_t____et_e_e____e_e___e_
```

Now you notice how there are a lot of "t_e" in the resulting text.
The most common trigram in the English alphabet is "the", so it's safe to say that "t_e" -> "the"
The first occurence of "t_e" is in the 7th character, which means our 'h' should be in the 8th character. If you look at the ciphertext, 's' is in our 8th char spot, and you can see how g is on the left of s, and v is on the right of s, the two chars we replaced, so let's try replacing s with h. This is the resulting text:

```
th____the_____te_tth_t___e_____te______the____t_t_t____________t___t_______h__e____e________e___he_e_t
he_e____e_____e__e_eth_t______h__e_ette____e_t________h__the_________t_t_t______he______thet_______e__
______e_____________t___e__e__ette________t_the____e_the_e_________e____________________he_e___e_te___
_the_e______e___e_te__e_t____ethe_e___he_e___e_
```

I chose "the" because this is the most common trigram in the english alphabet. Now let's look at the second most common trigram: "and". In our ngram program, it says the second most common trigram is "rmt". Let's try replacing "rmt" with "and". Here is the resulting text:

```
tha_a_the___ante_tth_t___en____te___andthe____tat_ta_n_n___nd__t___ta_n____h__e____e________e_a_he_e_t
he_e___de_n_a_e_ae_eth_t___n__h__e_ette__n_e_t_n_and_nh__the___a_____tat_ta_n_a_he______thet_a___n_e__
______e________and__t_n_e__e__ette______ant_the____e_the_e__anand_ne__a________a________he_ea__ente_an
dthe_e_nand_e___enten_e_t____ethe_e_a_he_e__ae_
```

However, this doesn't work because if you look at the first few letters, there are very little to no words that start with 'tha' and don't also have a 't', 'a', 'h', 'e', 'n', or 'd'. The only ones I can think of that could fit here, is "that", but it doesn't fit these conditions still. So let's get rid of "and" as rmt. Let's keep trying different bigrams. So we know that the is already in the resulting text. We know that 'g' maps to 't', 's' maps to 'h' and 'v' maps to 'e'. The next most common bigram in our ciphertext is 'vi'. Since we know that v = e, let's find the most common bigram in English alphabet that starts with e: "er". Let's try replacing 'i' with 'r'. This is the resulting text:

```
th____the_____te_tth_t___e__r__te______the____t_t_t__________r_t___t_______h__e____e________e___here_t
he_e____e_____e__e_eth_t______h__e_etter___ert________h__the_________t_t_t______her__r__thetr______err
_r_r__e_______r_____t___e__e__etter_______t_the____e_there_________e__________________r__here___e_ter__
_the_e______e___e_te__e_t____ethe_e___here___er
```

You can see how you can read a couple of "there"s and "here"s. I think this is correct now that we have a couple words other than "the". Now if we look at the most bigrams again, we get the table:

sv	gs	vi	rm
13	12	11	9

We know that "the" is "gsv", so we can essentially figure out:

he	th	e'i'	"rm"
13	12	11	9

We know that "in" is the next most frequently found bigram, so let's substitute "rm" with "in" because we know that the first letter of "vi" does not start with an 'i', but with an 'e'. This is the resulting text:

```
thi_i_the___inte_tth_t___en_r__te___in_the____tit_ti_n_n___n_r_t___ti_n____h__e____e________e_i_here_t
he_e____e_n_i_e_ie_eth_t___n__h__e_etter_n_ert_n_in__nh__the___i_____tit_ti_n_i_her__r__thetri___n_err
```

```
_r_r__e_______rin___t_n_e__e__etter_____int_the____e_there__inin__ne__i________i_____r__herei__enterin
_the_e_nin__e___enten_e_t____ethe_e_i_here__ier
```

Now we're getting somewhere. We have a lot of words we can make out now: "there", "here", "the", "in," and even "enter". Let's go back to unigrams. The most next unused most common letter is 'h' = 28. The most common letters in the english alphabet are: 'E', 'T', 'O', 'A', 'I', 'N', 'S', 'H', ... according to the slides. However, we already used 'E', 'T', 'N', 'I', and 'H' so we're left with 'O', 'A', 'S'. Let's plug each one of them for 'h' to see if anything else makes sense:
o: here we have words like "theo"

```
thioiothe___inte_tth_t__oen_r__te__oin_theo__otit_ti_n_n___n_r_t___ti_no___h__eo___eoo______e_i_here_t
he_eoo__e_n_i_e_ie_eth_t___n__h__e_etter_n_ert_n_in__nh__the__oi_o__otit_ti_n_i_her__r_othetri___n_err
_r_r__eoo__o__rin___t_n_e__e__ettero____int_the____eothere__inin__neo_i____o___i_____r__herei__enterin
_the_e_nin__eoooenten_eot____ethe_e_i_here_oier
```

a: here we have words like "eat"

```
thiaiathe___inte_tth_t__aen_r__te__ain_thea__atit_ti_n_n___n_r_t___ti_na___h__ea___eaa______e_i_here_t
he_eaa__e_n_i_e_ie_eth_t___n__h__e_etter_n_ert_n_in__nh__the__ai_a__atit_ti_n_i_her__r_athetri___n_err
_r_r__eaa__a__rin___t_n_e__e__ettera____int_the____eathere__inin__nea_i____a___i_____r__herei__enterin
_the_e_nin__eaaaenten_eat____ethe_e_i_here_aier
```

s: here we have words like "this", "is", "sentenc_e", "s__stit_tio_n"

```
thisisthe___inte_tth_t__sen_r__te__sin_thes__stit_ti_n_n___n_r_t___ti_ns___h__es___ess______e_i_here_t
he_ess__e_n_i_e_ie_eth_t___n__h__e_etter_n_ert_n_in__nh__the__si_s__stit_ti_n_i_her__r_sthetri___n_err
_r_r__ess__s__rin___t_n_e__e__etters____int_the____esthere__inin__nes_i____s___i_____r__herei__enterin
_the_e_nin__esssenten_est____ethe_e_i_here_sier
```

We can see that the letter that makes the most sense is most likely 's'.
We can make out a few words and infer that senten_e is likely sentence, and s__stit_ti_n is most likely substitution.
Let's make this happen by replacing the respective letters: x -> c, f -> u, y -> b. This is the resulting text:

```
thisisthe___inte_tth_t__sencr__te_usin_thesubstituti_n_n_c_n_r_tu__ti_ns__uh__esuccess_u____eci_here_t
he_ess__e_n_ibe_ie_eth_t__un__h__ebetterun_ert_n_in__nh__theb_sicsubstituti_nci_her__r_sthetri___n_err
_r_r_cess__sb_rin_but_nce__e__etters____int_the___cesthere__inin__nes_i____s__uic____r__herei__enterin
_the_e_nin__esssentencest____ethe_eci_here_sier
```

we can vaguely make out the words "substitution" "cipher", so we can infer that l -> o and k -> p. Let's try this and see what we can as the resulting text:

```
thisisthep__inte_tth_t__sencr_pte_usin_thesubstitution_n_con_r_tu__tions_ouh__esuccess_u____eciphere_t
he_ess__e_n_ibe_ie_eth_t_ouno_h__ebetterun_ert_n_in_onho_theb_sicsubstitutioncipher_or_sthetri___n_err
orprocess__sborin_butonce__e__etters____intothep__cesthere__inin_ones_i___oso_uic____ro_herei__enterin
_the_e_nin__esssentencesto___ethe_eciphere_sier
```

we can infer that "b_sic" is "basic", so let's replace the corresponding letter (z -> a). This is the resulting text:

```
thisisthep_ainte_tthat_asencr_pte_usin_thesubstitutionan_con_ratu_ations_ouha_esuccess_u____eciphere_t
he_essa_ean_ibe_ie_ethat_ouno_ha_ebetterun_ertan_in_onho_thebasicsubstitutioncipher_or_sthetria_an_err
orprocess_asborin_butoncea_e__etters_a__intothep_acesthere_ainin_ones_i___oso_uic____ro_hereia_enterin
_the_eanin__esssentencesto_a_ethe_ecipherеasier
```

We can see "encr*pye*" goes to encrypted. We can vaguely make out "con_ratu_ations" to congratulations. Therefore, b -> y, t -> g, and o -> l. This is the resulting text:

```
thisistheplainte_tthat_asencrypte_usingthesubstitutionan_congratulationsyouha_esuccess_ully_eciphere_t
he_essagean_ibelie_ethatyouno_ha_ebetterun_ertan_ingonho_thebasicsubstitutioncipher_or_sthetrialan_err
orprocess_asboringbutoncea_e_letters_allintotheplacesthere_ainingones_ill_oso_uic_ly_ro_hereia_enterin
gthe_eaninglesssentencesto_a_ethe_ecipher easier
```

You can make out a lot from this text.

```
thisistheplaintextthatwasencryptedusingthesubstitutionandcongratulationsyouhavesuccessfullydecipheredt
hemessageandibelievethatyounowhavebetterundertandingonhowthebasicsubstitutioncipherworksthetrialanderr
orprocesswasboringbutonceafewlettersfallintotheplacestheremainingoneswilldosoquicklyfromhereiamenterin
gthemeaninglesssentencestomakethedecipher easier
```

The message is certainly true. The more letters you figure out, the easier it becomes to decipher the message.