Assignment #1

20192973 김태현

Q1.

a. A student maintaining a blog to post public information.

Confidentiality: Low. Since the purpose of the blog is to post public information, there is no expectation of privacy or secrecy.

Availability: Moderate. Although the blog is for public information, availability matters because readers should be able to access the blog when they want. Any disruptions might affect the reputation or viewer counts of the blog.

Integrity: Moderate. Integrity is important because the blogger wouldn't want the content to be altered maliciously. Misinformation could harm the blogger's reputation.

b. An examination section of a university that manages sensitive information about exam papers.

Confidentiality: High. Exam papers are confidential until they are administered. Leaked exams could compromise the whole testing process.

Availability: High. If the examination system becomes unavailable, it might disrupt the entire examination schedule, causing delays and inconveniences. So it should prevents an unauthorized entity from making resources unavailable.

Integrity: High. Alteration in any exam paper's content can lead to misleading outcomes, which can deeply affect a student's future.

c. An information system in a pathological laboratory maintains the patient's data.

Confidentiality: High. Patient's data is sensitive and confidential. Unauthorized access could lead to misuse of data, identity theft, and could violate laws and regulations.

Availability: High. In a medical scenario, it's crucial to have timely access to patient data for treatments.

Integrity: High. Incorrect patient data might lead to incorrect treatments, which can be life-threatening.

d. A student information system used for maintaining student data in a university.

•Personal and academic information:

Confidentiality: High. This data is sensitive, and unauthorized disclosure could lead to various risks

including identity theft.

Availability: Moderate. While this information isn't needed on a continuous basis, it's important it's available when required for academic and administrative purposes.

Integrity: High. Inaccurate academic records can affect a student's progression, graduation, and future opportunities. Intentional or accidential data changes should be detectable.

•Routine administrative information (not privacy-related):

Confidentiality: Low. This information isn't privacy-related.

Availability: Low. A brief disruption might not cause major inconvenience.

Integrity: Moderate. While it's not privacy-related, wrong administrative data can cause confusion.

•Overall System:

Confidentiality: High.

Availability: Moderate.

Integrity: High.

e. A University library contains a library management system.

•Student data:

Confidentiality: High. Since student data is sensitive.

Availability: Moderate. Students need to borrow or return books, and this data should be available when needed.

Integrity: High. Incorrect data might lead to wrong penalties or missing books.

•Book data:

Confidentiality: Low. The details about the books are not private.

Availability: High. To efficiently manage the circulation of books, this data needs to be readily available.

Integrity: High. Incorrect book data can cause a lot of confusion in the library system.

•Overall System:

Confidentiality: Moderate.

Availability: High.

Integrity: High.

Q2.

1.Initial n bits of the plaintext contain a known pattern:

This type of redundancy is predictable and static. An attacker who knows the cryptographic scheme can use this known pattern as a way to check if a guessed decryption key is correct. If decryption with a guessed key and the attacker retrieves the known pattern, it's a strong indicator that the guessed key might be the correct one. This predictability makes known plaintext attacks feasible and might compromise the security of the encryption scheme.

2.Final n bits of the message contain a hash over the message:

Hash functions produce a fixed-size output (hash) based on the message. Even a small change in the input will produce a wildly different hash. This type of redundancy helps ensure the integrity of the message. If an attacker alters the message, the hash will change, indicating potential tampering. Plus, an attacker wouldn't be able to determine the correctness of a key based on the hash alone without having the original message for comparison.

From a security point of view, the two forms of redundancy are not equivalent:

The known pattern in the initial n bits introduces a vulnerability because it gives attackers a predictable target when trying decryption keys.

The hash in the final n bits provides a measure of integrity without offering the same level of predictability to attackers.

Therefore, from a security poinrt of view, having the final n bits of the message contain a hash over the message is a more secure form of redundancy compared to the initial n bits containing a known pattern.

Q3.

When using DES in ciphertext block chaining (CBC) mode, the decryption process involves taking a block of ciphertext, decrypting it, and then XORing the result with the previous block of ciphertext to produce the plaintext. Let's say P_i is the plaintext block in block C_i

1.Decryption of Block C_i : This block will be decrypted incorrectly because one of its bits has changed. After decryption, the result is XORed with the previous ciphertext block C_{i-1} to get the

plaintext block P_i. Since this process involves a bitwise XOR operation, the alteration in one bit of C_i will result in a corresponding single bit error in P_i.

2.Decryption of Block C_{i+1} : This block is decrypted and then XORed with the original C_i (which had the bit error) to produce the plaintext block P_{i+1}. Because the bitwise XOR operation is applied between C_{i+1} and C_i ,the single bit error in C_i will cause a single bit error in P_{i+1}.

In conclusion, one bit error in ciphertext block C_i will result in one bit being garbled in plaintext in block C_{i+1} as a result.

Q4.

Key:

Plaintext	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
Chipertext	q	p	x	b	n	k	y	z	s	?	m	r	g	f	l	h	?	j	c	o	e	d	w	a	v	?

In the ciphertext, the letters i, t, and u are missing. They likely match up with j, q, and z, but without more information, we can't tell which letter matches which.

Plain text:

phileas fogg was not known to have either wife or children,

which may happen to the most honest people; either relatives

or near friends, which is certainly more unusual. he lived

alone in his house in saville row, whither none penetrated.

a single domestic sufficed to serve him. he breakfasted and

dined at the club, at hours mathematically fixed, in the same

room, at the same table, never taking his meals with other

members, much less bringing a guest with him; and went home

at exactly midnight, only to retire at once to bed. he never

used the cosy chambers which the reform provides for its

favoured members. he passed ten hours out of the twenty-four

in saville row, either in sleeping or making his toilet.

Q5

Plaintext: explanation

Key: leg

1. Extend the key to match the length of the plaintext => legleglegle
2. Shift values for each letter in the key **l: 11 e: 4 g: 6**

Encryption

e (shift by 11) -> p

x (shift by 4) -> b

p (shift by 6) -> v

l (shift by 11) -> w

a (shift by 4) -> e

n (shift by 6) -> t

a (shift by 11) -> l

t (shift by 4) -> x

i (shift by 6) -> o

o (shift by 11) -> z

n (shift by 4) -> r

Encrypted text: pbvwetlxozr

Q6.

(a) Encrypt the plaintext "send more money" with the keystream:

a=0, b=1, …, z=25 (alphabetic mapping)

Plaintext: "send more money" Given Key stream: 9 0 1 7 23 15 21 14 11 11 2 8 9

s (shift by 9) -> b

e (shift by 0) -> e

n (shift by 1) -> o

d (shift by 7) -> k

m (shift by 23) -> j

o (shift by 15) -> d

r (shift by 21) -> m

e (shift by 14) -> s

m (shift by 11) -> x

o (shift by 11) -> z

n (shift by 2) -> p

e (shift by 8) -> m

y (shift by 9) -> h

ciphertext : beokjdmsxzpmh

(b) Using the ciphertext produced in part (a), find a key so that the ciphertext decrypts to the plaintext "cashnotneeded"

(b(1) – c(2))mod26 = 25

(e(4) – a(0))mod26 = 4

(o(14) – s(18))mod26 = 22

(k(10) – h(7))mod26 = 3

(j(9) – n(13))mod26 = 22

(d(3) – o(14))mod26 = 15

(m(12) – t(19))mod26 = 19

(s(18) – n(13))mod26 = 5

(x(23) – e(4))mod26 = 19

(z(25) – e(4))mod26 = 2

(p(15) – d(3))mod26 = 12

(m(12) – e(4))mod26 = 8

(h(7) – d(3))mod26 = 4

key stream : 25 4 22 3 22 15 19 5 19 21 12 8 4

Q7.

(1) Analyze and identify the ciphering technique used

dhtcgmicskfstadpnixklsduryojqjlmdiflrbppfcdbcztdbczptpipnvlmydhdddihsmtltigabtplkicbwoojgmmdpgiuehhq
oexhqaacsguggxeotxcdyocimmmthnlazxlnwdgbzoulgzddbjltxizlltigabtacaiiqcseixrieatrptqtuoihecywdgbzoulgzd
dbjltxizlacsuoeciflbgdilnwtyyttsnloeacsikxlniciflwdgjkavgcltwtplwxajiepcyusltpseixrieudpahdjeotwtwtanqcwag
icktwtplihhrplapaoacrcahpiroenlgslhtcahtgcdiaazlacplzwtgjltxizlltigabtacaiiqcseixrieatrptqtwlawiflrtlgslqtyuach
ulratrptqtjltxizlltigabtacaiiqcseixrielwgzptguvrshmmwxhbvmatrptqtjltxizlltigabtacaiiqcfepwjltxizlwwxqweglmy
dhdddihsmtltigabtplkwwtlahtcgnhixqjldjbftwtplihhrplapjpgwiroaihfpnthmumthfpntjlaiaimtoggmdltigabtxuhktjn
aoiwczojcbvfbjqpcbdroegbyyyrdklsidklsetyricvuvrshmmwxhbvmatrptqtjltxizlltigabtacaiiqcfepwjltxizltwtplwxaj
iepcyusltpseixrieatrptqtjltxizlltigabtnchhatrptqtroegtuplaqchnpcqdegacaiiqcseixrieatrptqtjltxizlytpfseixrielwgzpt
guvrshmmwxhbvmatrptqt

Given the length of the ciphertext and its consistent appearance without any spaces, this could be a Vigenère cipher. The presence of repeating sequences (like " zltxizl", "ltigabt", "ltigabt" and "acaiiqc", which appear multiple times) suggests the possibility of periodic key repetition, another hallmark of the Vigenère cipher.

(2) Identifying the correct key and the corresponding plaintext (1 point)

Let's say postion of first character of cipher text starts from 1.

zltxizl (reapeated sequences)

position : 150, 205, 365,430,485,510,695,720,760,825

differences between these positions : 55, 160, 65, 55, 25, 185, 25, 40, 65 => The greatest common divisor (GCD) : 5

ltigabt (reapeated sequences)

position : 72,157,372,437,492,537,617,702,767

differences between these positions : 85, 215, 65, 55, 45, 80, 85, 65 => The greatest common divisor (GCD) : 5

acaiiqc (reapeated sequences)

position : 164, 379, 444, 499, 804

differences between these positions : 215, 65, 55, 305 => The greatest common divisor (GCD) : 5

After measuring the distance between their repetitions. The greatest common divisor (GCD) of these distances are 5, might indicate the length of the key.

Now by using frequency analysis on every Nth character in each group to potentially derive the key

	A	B	C	D	E	F	G	H	I	J	K
1	a: 22	a: 11	a: 14	a: 11	a: 1	key :	h: 3	a: 11	p: 7	p: 8	y: 6
2	b: 0	b: 10	b: 3	b: 6	b: 7		i: 9	b: 10	q: 10	q: 9	z: 11
3	c: 0	c: 2	c: 11	c: 7	c: 25		j: 1	c: 2	r: 1	r: 1	a: 1
4	d: 6	d: 5	d: 11	d: 7	d: 4		k: 4	d: 5	s: 3	s: 6	b: 7
5	e: 0	e: 26	e: 2	e: 1	e: 2		l: 39	e: 26	t: 34	t: 31	c: 25
6	f: 5	f: 1	f: 0	f: 0	f: 7		m: 6	f: 1	u: 3	u: 1	d: 4
7	g: 0	g: 2	g: 8	g: 11	g: 18		n: 3	g: 2	v: 2	v: 1	e: 2
8	h: 3	h: 7	h: 9	h: 15	h: 0		o: 11	h: 7	w: 10	w: 5	f: 7
9	i: 9	i: 18	i: 18	i: 28	i: 3		p: 18	i: 18	x: 17	x: 14	g: 18
10	j: 1	j: 0	j: 3	j: 5	j: 15		q: 0	j: 0	y: 0	y: 0	h: 0
11	k: 4	k: 1	k: 1	k: 0	k: 4		r: 1	k: 1	z: 1	z: 0	i: 3
12	l: 39	l: 15	l: 4	l: 6	l: 7		s: 10	l: 15	a: 14	a: 11	j: 15
13	m: 6	m: 6	m: 0	m: 0	m: 12		t: 4	m: 6	b: 3	b: 6	k: 4
14	n: 3	n: 6	n: 2	n: 1	n: 3		u: 6	n: 6	c: 11	c: 7	l: 7
15	o: 11	o: 9	o: 0	o: 0	o: 0		v: 7	o: 9	d: 11	d: 7	m: 12
16	p: 18	p: 3	p: 7	p: 8	p: 8		w: 2	p: 3	e: 2	e: 1	n: 3
17	q: 0	q: 0	q: 10	q: 9	q: 6		x: 0	q: 0	f: 0	f: 0	o: 0
18	r: 1	r: 6	r: 1	r: 1	r: 23		y: 7	r: 6	g: 8	g: 11	p: 8
19	s: 10	s: 6	s: 3	s: 6	s: 0		z: 10	s: 6	h: 9	h: 15	q: 6
20	t: 4	t: 27	t: 34	t: 31	t: 0		a: 22	t: 27	i: 18	i: 28	r: 23
21	u: 6	u: 0	u: 3	u: 1	u: 8		b: 0	u: 0	j: 3	j: 5	s: 0
22	v: 7	v: 0	v: 2	v: 1	v: 0		c: 0	v: 0	k: 1	k: 0	t: 0
23	w: 2	w: 11	w: 10	w: 5	w: 3		d: 6	w: 11	l: 4	l: 6	u: 8
24	x: 0	x: 0	x: 17	x: 14	x: 0		e: 0	x: 0	m: 0	m: 0	v: 0
25	y: 7	y: 2	y: 0	y: 0	y: 6		f: 5	y: 2	n: 2	n: 1	w: 3
26	z: 10	z: 0	z: 1	z: 0	z: 11		g: 0	z: 0	o: 0	o: 0	x: 0

First I just assumed i=>e, t=>e, t=>e,t=>e, c=>e since it appeared most in each group.

Word “hpppy” didn’t make sence, but I could easily guess the word “happy” from it. So changed e=>e, in group 2. After trying decryption with key “happy”, plain text made much sence so I could know that “happy” was the key.

Key : happy

(3) Programming code for decryption (3 points)

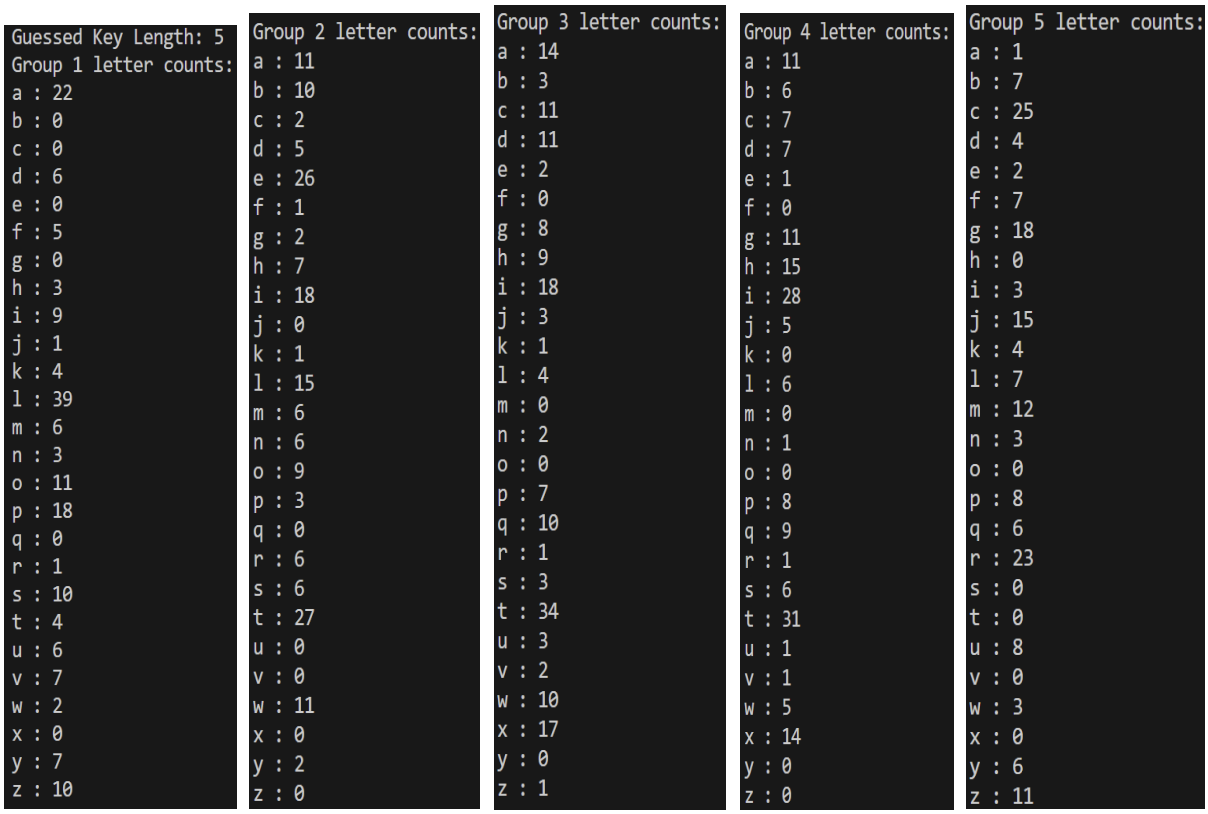
Code :

```
import math
from collections import defaultdict

ciphertext =
"dhtcgmicskfstadpnixklsduryojqjlmdiflrbppfcdbcztdbczptpipnvlmydhdddihsmtltigab
tplkicbwoojgmmdpgiuehhqoexhqaacsguggxeotxcdyocimmmthnlazxlnwdgbzoulgzddbjltxiz
lltigabtacaiiqcseixrieatrptqtuoihecywdgbzoulgzddbjltxizlacsuoeciflbgdilnwtyytt
snloeacsikxlniciflwdgjkavgcltwtplwxajiepcyusltpseixrieudpahdjeotwtwtanqcwagick
twtplihhrplapaoacrcahpiroenlgslhtcahtgcdiaazlacplzwtgjltxizlltigabtacaiiqcseix
rieatrptqtwlawiflrtlgslqtyuachulratrptqtjltxizlltigabtacaiiqcseixrielwgzptguvr
shmmwxhbvmatrptqtjltxizlltigabtacaiiqcfepwjltxizlwwxqweglmydhdddihsmtltigabtpl
kwwtlahtcgnhixqjldjbftwtplihhrplapjpgwiroaihfpnthmumthfpntjlaiaimtoggmdltigabt
xuhktjnaoiwczojcbvfbjqpcbdroegbyyyrdklsidklsetyricvuvrshmmwxhbvmatrptqtjltxizl
ltigabtacaiiqcfepwjltxizltwtplwxajiepcyusltpseixrieatrptqtjltxizlltigabtnchhat
rptqtroegtuplaqchnpcqdegacaiiqcseixrieatrptqtjltxizlytpfseixrielwgzptguvrshmmw
xhbvmatrptqt"

#key length guessing

# For each repeating sequence, calculate the distances between occurrences.
# For each distance, calculate its factors.
# Count how often each factor appears.
# The most common factor, or one of the most common factors, is likely the key
length.

def find_repeated_sequences(text, seq_length=3):
    seq_positions = {}
    for i in range(len(text) - seq_length):
        seq = text[i:i+seq_length]
        if seq in seq_positions:
            seq_positions[seq].append(i)
        else:
            seq_positions[seq] = [i]
    return {seq: positions for seq, positions in seq_positions.items() if
len(positions) > 1}

def get_factors(n):
    factors = set()
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            factors.add(i)
            factors.add(n // i)
```

```
    return factors

def guess_key_length(ciphertext):
    repeated_seq = find_repeated_sequences(ciphertext)
    distances = []
    for seq, positions in repeated_seq.items():
        for i in range(1, len(positions)):
            distances.append(positions[i] - positions[i-1])

    factor_counts = defaultdict(int)
    for distance in distances:
        for factor in get_factors(distance):
            factor_counts[factor] += 1

    # Return the factor that appears most often, excluding 1
    likely_key_length = max(factor_counts, key=factor_counts.get)

    return likely_key_length

key_length_guess = guess_key_length(ciphertext)
print(f"Guessed Key Length: {key_length_guess}")

#frequency analysis

def count_letters(group):
    count = {}
    for char in 'abcdefghijklmnopqrstuvwxyz':
        count[char] = group.count(char)
    return count

def separate_into_groups(ciphertext, key_length):
    groups = [""] * key_length

    for i, char in enumerate(ciphertext):
        groups[i % key_length] += char

    return groups

groups = separate_into_groups(ciphertext, key_length_guess)

for i, group in enumerate(groups, 1):
    letter_counts = count_letters(group)
    print(f"Group {i} letter counts:")
    for char, count in letter_counts.items():
        print(f"{char} : {count}")
    print()

def vigenere_decrypt(ciphertext, key):
```

```
    decrypted = ""
    key_length = len(key)
    alphabet = "abcdefghijklmnopqrstuvwxyz"

    for i in range(len(ciphertext)):
        char = ciphertext[i]
        if char.isalpha():  # decrypt only alphabets
            shift = alphabet.index(key[i % key_length].lower())
            decrypted += alphabet[(alphabet.index(char.lower()) - shift) % 26]
        else:
            decrypted += char

    return decrypted

#make the most frequent letter into 'e'
def get_shift_for_most_frequent_to_e(group):
    alphabet = "abcdefghijklmnopqrstuvwxyz"

    # Get the most frequent letter in the group
    letter_counts = count_letters(group)
    most_frequent_letter = max(letter_counts, key=letter_counts.get)

    # Calculate the shift needed to make the most frequent letter into 'e'
    shift = (ord(most_frequent_letter) - ord('e')) % 26
    return shift

key = ""

alphabet = "abcdefghijklmnopqrstuvwxyz"
for group in groups:
    shift = get_shift_for_most_frequent_to_e(group)
    key += alphabet[shift]

print("Predicted Key:", key) #didn't make sence.

plaintext = vigenere_decrypt(ciphertext, key)
print("Plaintext:", plaintext)

print()

plaintext_answer = vigenere_decrypt(ciphertext, "happy")
print("Plaintext:", plaintext_answer)
```

Output :


Guessed Key Length: 5
Group 1 letter counts:
a : 22
b : 0
c : 0
d : 6
e : 0
f : 5
g : 0
h : 3
i : 9
j : 1
k : 4
l : 39
m : 6
n : 3
o : 11
p : 18
q : 0
r : 1
s : 10
t : 4
u : 6
v : 7
w : 2
x : 0
y : 7
z : 10
Group 2 letter counts:
a : 11
b : 10
c : 2
d : 5
e : 26
f : 1
g : 2
h : 7
i : 18
j : 0
k : 1
l : 15
m : 6
n : 6
o : 9
p : 3
q : 0
r : 6
s : 6
t : 27
u : 0
v : 0
w : 11
x : 0
y : 2
z : 0
Group 3 letter counts:
a : 14
b : 3
c : 11
d : 11
e : 2
f : 0
g : 8
h : 9
i : 18
j : 3
k : 1
l : 4
m : 0
n : 2
o : 0
p : 7
q : 10
r : 1
s : 3
t : 34
u : 3
v : 2
w : 10
x : 17
y : 0
z : 1
Group 4 letter counts:
a : 11
b : 6
c : 7
d : 7
e : 1
f : 0
g : 11
h : 15
i : 28
j : 5
k : 0
l : 6
m : 0
n : 1
o : 0
p : 8
q : 9
r : 1
s : 6
t : 31
u : 1
v : 1
w : 5
x : 14
y : 0
z : 0
Group 5 letter counts:
a : 1
b : 7
c : 25
d : 4
e : 2
f : 7
g : 18
h : 0
i : 3
j : 15
k : 4
l : 7
m : 12
n : 3
o : 0
p : 8
q : 6
r : 23
s : 0
t : 0
u : 8
v : 0
w : 3
x : 0
y : 6
z : 11

```
Predicted Key: hpppy
Plaintext: wseniftndmydelfiytimedoftrzublexothecmarynomeseomesaeakiygworosofwtsdomwetitmeandtnmyhzurofoarknpssshpisstlndinrrig
einfrzntofxespelkinghordszfwisoomleeitbewetitmeletttbelptitbpletiebewhtsperhordszfwisoomleeitbelndwhpnthemrokeyheareedpezpleltv
ingtnthehorldlgreeeherehillbpanandwerlptitbpfortsougheheymlybeplrtedeheretsstiwlachlncetsatthpywilwseetserewtllbelnansherleeitb
ewetitmeletttbelptitbpletiebeyelhthecewilwbeanlnswecletiebeleeitbewetitmeletttbelptitbpwhisaerwocdsofhisdoxletiebeleeitbewetitn
eletttbeypahleeitbehhispprworosofwtsdomwetitmeandhhentsenigstiscwoudyeheretsstiwlalirhtthltshiyesonxeshiyeunttltomzrrowwetitmei
waveuptztheszundoqmusinmothprmarjcomedtomedpeaktngwocdsofhisdoxletiebeleeitbewetitmeletttbeypahleeitbeeherehillbpanandwerlptitb
pletiebeleeitbewetitmeyeasletiebethprewiwlbeayanswprletttbelptitbpletiebeleeitbejeahlptitbpwhisaerwocdsofhisdoxletiebe

Plaintext: whenifindmyselfintimesoftroublemothermarycomestomespeakingwordsofwisdomletitbeandinmyhourofdarknesssheisstandingrigh
tinfrontofmespeakingwordsofwisdomletitbeletitbeletitbeletitbeletitbewhisperwordsofwisdomletitbeandwhenthebrokenheartedpeopleliv
inginthe worldagreetherewillbeananswerletitbeforthoughtheymaybepartedthereisstillachancethattheywillseetherewillbeananswerletitb
eletitbeletitbeletitbeletitbeyeahtherewillbeananswerletitbeletitbeletitbeletitbeletitbewhisperwordsofwisdomletitbeletitbeletitb
eletitbeyeahletitbewhisperwordsofwisdomletitbeandwhenthenightiscloudythereisstillalightthatshinesonmeshineuntiltomorrowletitbe
wakeuptothesoundofmusicmothermarycomestomespeakingwordsofwisdomletitbeletitbeletitbeletitbeyeahletitbetherewillbeananswerletitb
eletitbeletitbeletitbeyeahletitbetherewillbeananswerletitbeletitbeletitbeletitbeyeahletitbewhisperwordsofwisdomletitbe
PS C:\Users\USER\Desktop>
```