Assignment 3

● Graded

Student

Sivachandran P

Total Points

70 / 100 pts

Question 1

Commands

10 / 10 pts

Solved in the server

Question 2

Cryptosystem

5 / 5 pts

The rubric is hidden for this question.

Question 3

Analysis

50 / 80 pts

The rubric is hidden for this question.

Question 4

Password

5 / 5 pts

The rubric is hidden for this question.

Question 5

Code

0 / 0 pts

The rubric is hidden for this question.

Q1 Commands

10 Points

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

go -> enter -> pluck -> c -> c -> back -> give -> back -> back -> thrnxxtzy -> read

Q2 Cryptosystem

5 Points

What cryptosystem was used in this level?

Transposition(Permutation)-Substitution Cipher

Q3 Analysis

80 Points

What tools and observations were used to figure out the cryptosystem and the password? (Explain in less than 1000 lines)

Tool and observations used:
Written python code to decrypt the cipher text found in this assignment.
Listing down the steps involved in this program:

1) Capture the letter counts found in cipher text
2) Analyze the letter's Frequency distribution in the cipher text
3) Check the spaces and special characters from this cipher text and remove
4) Prepare Cipher text in a group of 5 letters
5) Preparation of permutation for a set of elements {0, 1, 2, 3, 4}
6) Arriving permuted text considering permutation of order 5 {3, 2, 4, 0, 1}
7) According to cipher text order shuffle the letters at Substituted text
8) Verify the decrypted cipher text with terminal

Sharing below my observations followed:
1) From terminal found the following cipher text:

```
qmnjvsa nv wewc flct vprj tj tvvplvl fv xja vqildhc
xmlnvc nacyclpa fc gyt vfvw. fv wgqyp, pqq pqcs y wsq
rx qmnjvafy cgv tlvhf cw tyl aeuq fv xja tkbv cqnsqs.
lhf avawnc cv eas fuqb qvq tc yllrqr xxwa cfy. psdc uqf
avrqc gefq pyat trac xwv taa wwd dv eas flcbq. vd trawm
vupq quw x decgqcwt, yq yafl vlqs yqklhq! snafq vml
lhvqpawr nqg_vfusr_ec_wawy qp fn wgawdgf.
```

2) Here looks like every alphabet got replaced with different letter, and may be it could hint that substitution cipher. Planned to proceed with analyze Frequency of the English letters, accordingly removed both empty spaces and special characters found at cipher text. Sharing below the analysis result:
q -> 10.56%
v -> 10.21%
a -> 8.10%
c -> 7.75%
f -> 6.69%
w -> 6.69%
l -> 5.99%
t -> 4.58%
y -> 4.58%

p -> 3.87%
s -> 3.87%
n -> 3.52%
r -> 3.17%
g -> 2.82%
x -> 2.82%
d -> 2.46%
e -> 2.46%
j -> 2.11%
u -> 2.11%
h -> 1.76%
m -> 1.76%
b -> 1.06%
k -> 0.70%
i -> 0.35%

3) Above analysis tells that "q", "v", "f", "c" letters are noticed frequently in the above cipher text. In addition to it, noticed that there are occurrence of double letters such as "vv", "qq", "ww". In general substitution cipher, the double occurrence of letters might not translate to double letters in the cipher text. Apart from this observed that letters "x" & "y" also in the cipher text. It hints me that it should not be the type of simple Caesar cipher type. Accordingly tried to check with Transposition-Substitution Cipher.

4) From the cipher text provided, since 'nqg_vfusr_ec_wawy' letters are linked with "_", I assume it may be the password, based on the hint observed in the previous level. One another obersvation found is 'vml' could be the word 'the' according to previous level password decryption.

5) As next step of analysis, removed the password text ('nqg_vfusr_ec_wawy') from the cipher and now found 270 letters to be analyzed (out of total 284 letters). The purpose of removing password text 'nqg_vfusr_ec_wawy' is to divide the new cipher text length, into block of equal letters. 2,3,5 and 10 are divisible to 270. As I have mentioned above randomly assumed the block length as 5.

6) To implement the above step 5, special characters and empty spaces from the cipher text are removed and divided it into block or group of 5. Accordingly found the following text:

qmnjv sanvw ewcfl ctvpr jtjtv vplvl fvxja vqild hcxml nvcna cyclp afcgy tvfvw fvwgq yppqq
pqcsy wsqrx qmnjv afycg vtlvh fcwty laeuq fvxja tkbvc qnsqs lhfav awncc veasf uqbqv qtcyl lrqrx

xwacf ypsdc uqfav rqcge fqpya ttrac xwvta awwdd veasf lcbqv dtraw mvupq quwxd ecgqc wtyqy
aflvl qsyqk lhqsn afqvm llhvq pawrn qgvfu srecw awyqp fnwga wdgf

7) Block/group length considered is 5, the key should be permitted within {0, 1, 2, 3, 4}. After performing permutation of {0, 1, 2, 3, 4} would be 5! options. It resulted the following 120 possible permutations:

(0, 1, 2, 3, 4) (0, 1, 2, 4, 3) (0, 1, 3, 2, 4) (0, 1, 3, 4, 2) (0, 1, 4, 2, 3)
(0, 1, 4, 3, 2) (0, 2, 1, 3, 4) (0, 2, 1, 4, 3) (0, 2, 3, 1, 4) (0, 2, 3, 4, 1)
(0, 2, 4, 1, 3) (0, 2, 4, 3, 1) (0, 3, 1, 2, 4) (0, 3, 1, 4, 2) (0, 3, 2, 1, 4)
(0, 3, 2, 4, 1) (0, 3, 4, 1, 2) (0, 3, 4, 2, 1) (0, 4, 1, 2, 3) (0, 4, 1, 3, 2)
(0, 4, 2, 1, 3) (0, 4, 2, 3, 1) (0, 4, 3, 1, 2) (0, 4, 3, 2, 1) (1, 0, 2, 3, 4)
(1, 0, 2, 4, 3) (1, 0, 3, 2, 4) (1, 0, 3, 4, 2) (1, 0, 4, 2, 3) (1, 0, 4, 3, 2)
(1, 2, 0, 3, 4) (1, 2, 0, 4, 3) (1, 2, 3, 0, 4) (1, 2, 3, 4, 0) (1, 2, 4, 0, 3)
(1, 2, 4, 3, 0) (1, 3, 0, 2, 4) (1, 3, 0, 4, 2) (1, 3, 2, 0, 4) (1, 3, 2, 4, 0)
(1, 3, 4, 0, 2) (1, 3, 4, 2, 0) (1, 4, 0, 2, 3) (1, 4, 0, 3, 2) (1, 4, 2, 0, 3)
(1, 4, 2, 3, 0) (1, 4, 3, 0, 2) (1, 4, 3, 2, 0) (2, 0, 1, 3, 4) (2, 0, 1, 4, 3)
(2, 0, 3, 1, 4) (2, 0, 3, 4, 1) (2, 0, 4, 1, 3) (2, 0, 4, 3, 1) (2, 1, 0, 3, 4)
(2, 1, 0, 4, 3) (2, 1, 3, 0, 4) (2, 1, 3, 4, 0) (2, 1, 4, 0, 3) (2, 1, 4, 3, 0)
(2, 3, 0, 1, 4) (2, 3, 0, 4, 1) (2, 3, 1, 0, 4) (2, 3, 1, 4, 0) (2, 3, 4, 0, 1)
(2, 3, 4, 1, 0) (2, 4, 0, 1, 3) (2, 4, 0, 3, 1) (2, 4, 1, 0, 3) (2, 4, 1, 3, 0)
(2, 4, 3, 0, 1) (2, 4, 3, 1, 0) (3, 0, 1, 2, 4) (3, 0, 1, 4, 2) (3, 0, 2, 1, 4)
(3, 0, 2, 4, 1) (3, 0, 4, 1, 2) (3, 0, 4, 2, 1) (3, 1, 0, 2, 4) (3, 1, 0, 4, 2)
(3, 1, 2, 0, 4) (3, 1, 2, 4, 0) (3, 1, 4, 0, 2) (3, 1, 4, 2, 0) (3, 2, 0, 1, 4)
(3, 2, 0, 4, 1) (3, 2, 1, 0, 4) (3, 2, 1, 4, 0) (3, 2, 4, 0, 1) (3, 2, 4, 1, 0)
(3, 4, 0, 1, 2) (3, 4, 0, 2, 1) (3, 4, 1, 0, 2) (3, 4, 1, 2, 0) (3, 4, 2, 0, 1)
(3, 4, 2, 1, 0) (4, 0, 1, 2, 3) (4, 0, 1, 3, 2) (4, 0, 2, 1, 3) (4, 0, 2, 3, 1)
(4, 0, 3, 1, 2) (4, 0, 3, 2, 1) (4, 1, 0, 2, 3) (4, 1, 0, 3, 2) (4, 1, 2, 0, 3)
(4, 1, 2, 3, 0) (4, 1, 3, 0, 2) (4, 1, 3, 2, 0) (4, 2, 0, 1, 3) (4, 2, 0, 3, 1)
(4, 2, 1, 0, 3) (4, 2, 1, 3, 0) (4, 2, 3, 0, 1) (4, 2, 3, 1, 0) (4, 3, 0, 1, 2)
(4, 3, 0, 2, 1) (4, 3, 1, 0, 2) (4, 3, 1, 2, 0) (4, 3, 2, 0, 1) (4, 3, 2, 1, 0)

8) Column positions {0, 1, 2, 3, 4} refers to the positions within each block. First char: 0, Last char: 4. Based on the key column positions are shuffled at each block of the message encrypted. Ofcourse Key is permutation of the numbers 0 to 4. A permutation is an arrangement of all elements in a set (in our case {0,1,2,3,4}) says that no element appears twice or more

9) Remember the above point #6, divided the text into block/group of 5. In the same point noticed that word 'llhvq'. By hit and trial option tried with all the possible options to do manual permutation and accordingly found the option that {3, 2, 4, 0, 1} could be the order of permutation should be tried

while referring 'llhvq', then while applying {3, 2, 4, 0, 1} converts that 'vhqll', here while using the substitution 'h' : 'p', 'q' : 'a' & 'll': 'ss', helped to arrive the word 'vpass', it implies to arrive the word 'pass' (half word of 'password').

According to it the transposition key {3, 2, 4, 0, 1} to the entire cipher text, and it helped to arrive the following:
jnvqmv n wsaf clewp vrctt j vjtv llvpj x afvlidvqm xlhcn canvlcpcyg c yafv f.w tvg wqfvq,p q yps c
ypqr q xwsjnvqmc ygafv lhvtt w yfcu eqlaj x afvv bctkq.s sqna fvlhc ncaws a fveq bvuqy c lqtr
qxlrc afxwd.s c ypa fvuqg cerqy pafqa r cttt vaxwd w daws a fveq.b vlca rwdtp uqmvx w
dquqgcecq,y y wtv llafq ykqss!q nlhv qmafv hqllr wnpaf_vuqgc_e_wsrq y pawg wafnf.gwd

10) Now arrive the substitution method and substitution with the reference of frequency analysis and mapped like below:

'a': 'T', 'b': 'b', 'c': 'L', 'd': 'd',
'e': 'C', 'f': 'H', 'g': 'G', 'h': 'P'
'i': 'i', 'j': 'j', 'k': 'k', 'l': 'S'
'm': 'K', 'n': 'R', 'p': 'D', 'q': 'A'
'r': 'W', 's': 'F', 't': 't', 'u': 'u'
'v': 'E', 'w': 'O', 'x': 'x', 'y': 'y'

11) After the implementation of the above mentioned substitution, found the cipher text like below:

jreake r ofth lscod ewltt j ejte ssedj x thesideak xsplr ltresldlyg l ythe h.o teg oahea,d a ydf l
ydaw a xofjreakl ygthe spett o yhlu c astj x thee bltka.f fart hespl rltof t heca beuay l satw axswl
thxod.f l ydt heuag lcway dthat w lttt etxod o dtof t heca.b eslt wodtd u akex o dauaglcla,y y ote
sstha ykaff!a rspe akthe passw ordth_euagl_c_ofwa y dtog othrh.god

12) While tried to apply the substitution with simple below 11 letters:

'j': 'B', 't': 'L', 'x': 'Y', 'i': 'Q'
'd': 'U', 'y': 'N', 'l': 'T', 'b': 'V'
'k': 'J', 'u': 'M', 'L': 'T'

Now the substituted text will be like below:

breake r ofth iscod ewill b eble ssedb y thesqueak yspir itresiding i nthe h.o leg oahea,d a ndf i
ndaw a yofbreaki ngthe spell o nhim c astb y thee vilja.f fart hespi ritof t heca veman i salw
ayswi thyou.f i ndt hemag icwan dthat w illl etyou o utof t heca.v esit would m akey o
uamagicia,n n ole sstha njaff!a rspe akthe passw ordth_emagi_c_ofwa n dtog othrh.gou

13) As next step rearranged the word as above substituted text. While analyzing the first word of the original cipher text contains 7 letters, so the first word of decrypted text should also contain 7 letters. Then about the next word of cipher text contains 2 letters, accordingly second word of the decrypted text should rely 2 letters and son on...

14) After shuffle the substituted text got some more improved decrypted text like below:

breaker of this code will be blessed by the squeaky spirit residing in the h.ole go ahea,d and
find a way of breaking the spell on him cast by the evil ja.ffar the spirit of the cave man is always
with you. find the magic wand that will let you out of the ca.ves it would make you a magicia,n
no less than jaff!ar speak the password th_emagi_c_ofwand to go thrh.gou

15) While looking deep in to the decrypted text found some comma and fullstops. For example:

'h.ole' most likely to be 'hole.'
'ahea,d' most likely to be 'ahead,'
'thrh.gou' most likely to be 'through'

Now we could get more fine-grained decrypted text like below:

breaker of this code will be blessed by the squeaky spirit residing in the hole. go ahead, and
find a way of breaking the spell on him cast by the evil jaffar the spirit of the cave man is always
with you. find the magic wand that will let you out of the ca.ves it would make you a magician,
no less than jaffar! speak the password the_magic_of_wand to go through.

16) Now we arrived the final password which is required to clear this particular level: 'the_magic_of_wand'

Q4 Password

5 Points

What was the final command used to clear this level?

the _ magic _ of _ wand

Q5 Code

0 Points

Upload any code that you have used to solve this level.

Assignment_3.py Download

```python
from itertools import permutations


 #Function to find distinct letter count in cipher text
 #def count_letters_in_cipherText(cipherText):
def count_letters_in_cipherText(cipherText):
    letter_counts = {}
    for char in cipherText.lower():
        if char.isalpha():
            letter_counts[char] = letter_counts.get(char, 0) + 1
            return letter_counts

 #Function to find frequency distribution of letters in cipher text
def get_letter_frequency_distribution(cipherText):
        freq = {}
        for c in cipherText:
            if(c.isalpha()):
                lower_char = c.lower()
                freq[lower_char] = freq.get(lower_char,0)+1
                return freq

 #Function to remove spaces and special character from cipher text
def remove_spaces_from_cipher(cipherText):
    new_cipher = ""
    for c in cipherText:
        if(c.isalpha()):
            new_cipher += c
            return new_cipher

 #Function to divide cipher text into a block of 5
def divide_cipher_into_5block(new_cipher_text):
    block_cipher =""
    for i in range(0, len(new_cipher_text), 5):
        c = new_cipher_text[i:i+5]

        block_cipher += c + " "
        return block_cipher


 #Function to generate permutation for a set of elements {0,1,2,3,4}
def generate_permutations(elements):
    for permutation in permutations(elements):
        yield permutation

 #Function to get pertmuted text considering permutation of order 5 {3,2,4,0,1}
def get_permuted_text(cipherText,cipherlength):
    permuted_text = ""
    char_index = 0
    while char_index < cipherlength:
        try:
            chars = []
            separators = []
            for _ in range(5):
                if char_index < cipherlength and cipherText[char_index].isalpha():
                    chars.append(cipherText[char_index])
                    char_index += 1
                    while char_index < cipherlength and not cipherText[char_index].isalpha():
                        separators.append(cipherText[char_index])
                    char_index += 1
                else:
                    chars.append("")
                    if(char_index < cipherlength):
                                separators.append(cipherText[char_index])
                                char_index += 1

                                permuted_text += "".join([chars[3]] + separators[:1] + [chars[2]] +
                                                        separators[1:2] + [chars[4]] + separators[2:3] + [chars[0]] + separators[3:4] + [chars[1]]
                                                        + separators[4:])
        except IndexError:
            break
    return permuted_text

 #Function to substitute a character based with the corresponding letter mentioned in mapping
def substitute_mapping(char, mapping):
    return mapping.get(char, char)

 #Function to reorder the letters in decrypted text.
 #In cipher text first word contains 7 letter, so in
 #decrypted text first word should also contain 7 letter
 #and the same goes for other words in cipher and decrypted text
def re_order_letters(cipherText,permutedtext):
        pos = 0
        decryptedText = ""
        length =len(cipherText.split(' '));
        for l in range(length):
            s = cipherText.split(' ')[l]

            for i in range(len(s)):
                sl = len(s)
                if(permutedtext[pos].endswith("\n")): break
                if(permutedtext[pos] == ' '):
                        pos+=1
                        decryptedText += permutedtext[pos]
                        pos += 1
                        decryptedText += ' '
            return decryptedText

 # main function
def main():
    #cipherText = ("qmnjvsa nv wewc flct vprj tj tvvplvl fv xja vqildhc xmlnvc nacyclpa fc "
    #"gyt vfvw. fv wgqyp, pqq pqcs y wsq rx qmnjvafy cgv tlvhf cw tyl aeuq fv xja tkbv cqnsqs."
    #"lhf avawnc cv eas fuqb qvq tc yllrqr xxwa cfy. psdc uqf avrqc gefq pyat trac xwv taa wwd dv"
    #"eas flcbq. vd trawm vupq quw x decgqcwt, yq yafl vlqs yqklhq! snafq vml lhvqpawr"
    #"nqg_vfusr_ec_wawy qp fn wgawdgf.")

    cipherText = ("qmnjvsa nv wewc flct vprj tj tvvplvl fv xja vqildhc xmlnvc nacyclpa fc"
"gyt vfvw. fv wgqyp, pqq pqcs y wsq rx qmnjvafy cgv tlvhf cw tyl aeuq fv xja tkbv cqnsqs."
"lhf avawnc cv eas fuqb qvq tc yllrqr xxwa cfy. psdc uqf avrqc gefq pyat trac xwv taa wwd dv"
"eas flcbq. vd trawm vupq quw x decgqcwt, yq yafl vlqs yqklhq! snafq vml lhvqpawr"
"nqg_vfusr_ec_wawy qp fn wgawdgf.")
    cipherlen = len(cipherText)
    print(cipherlen)

# code to find new length of cipher text after removing spaces and cipher text
new_cipherlen =0;
for i in range(cipherlen):
     if(cipherText[i].isalpha()):
          new_cipherlen+=1

#code to return distinct letter count in cipher text
#and sort it in alphabetical order
letter_counts = count_letters_in_cipherText(cipherText)
sorted_counts = dict(sorted(letter_counts.items()))

for letter, count in sorted_counts.items():
    print(f"{letter} --> {count}")

#code to return Frequency Distribution of letters in cipher text
letter_frequency = get_letter_frequency_distribution(cipherText)
print("\nFrequency Distribution of letters in cipher text\n")

for c in range(26):
    l = chr(ord('a')+c)
    if(letter_frequency.get(l,0))>0:
        p = (letter_frequency.get(l,0)/new_cipherlen * 100)
        print(f"{l} --> {p:.4f}%")

 #code for returning cipher text after removing spaces and special character
new_cipher_text = remove_spaces_from_cipher(cipherText)
print(f"\nCipher after removing spaces and special character:\n{new_cipher_text}")

 # Code for returning cipher divided into a block of 5
cipher_into_block = divide_cipher_into_5block(new_cipher_text)
print(f"\nCipher in block/group of 5:\n{cipher_into_block}")
# code to print permutation of elements = [0, 1, 2, 3, 4]
```

```
elements = [0, 1, 2, 3, 4]
num_of_permutations = 24
permutation_generator = generate_permutations(elements)
print(f"\nAll permutations of {0, 1, 2, 3, 4}:".format(elements))

for _ in range(num_of_permutations):
    for i in range(5):
        get_permutation = next(permutation_generator)
        print(get_permutation, end=" ")
        print()


 # code to invoke function to get permuted text
permutedtext = get_permuted_text(cipherText,cipherlen)
print(f"\nPermuted Text:\n{permutedtext}")

 # Code of defining mapping
substitution_map = {
 "a": "T","b": "b","c": "L","d": "d","e": "C","f": "H","g": "G","h": "P","i": "i","j":
"j","k": "k","l": "S",
 "m": "K","n": "R","p": "D","q": "A","r": "W","s": "F","t": "t","u": "u","v": "E","w":
"O","x": "x","y": "y"
 }

substituted_text = "".join(substitute_mapping(char, substitution_map) for char in permutedtext)

print(f"\nText after First Substitution:\n{substituted_text.lower()}")

 # Code of defining mapping
substitution_map1 = {
"j":"B","t":"L","x":"Y","i":"Q","d":"U","y":"N","l":"I","b":"V","k":"J","u":"M","L":"I" }

substituted_text = "".join(substitute_mapping(char, substitution_map1) for char in substituted_text)

print(f"\nText after Second Substitution:\n{substituted_text.lower()}")

substituted_text +="\n"

decryptedText = re_order_letters(cipherText,substituted_text)

print("\n-------Decrypted Text After Re-ordering-------")
print(f"{decryptedText.lower()}")

if __name__ == "__main__":
    main()
```