Q1 Commands
5 Points

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

go
enter
read

Q2 Cryptosystem
5 Points

What cryptosystem was used in this level?

Substitution Cipher (Mono-Alphabetic)

Q3 Analysis
25 Points

What tools and observations were used to figure our the cryptosystem? (Explain in less than 100 words)

Tools:

1. We have used python script to check if the given ciphertext is encrypted with "SHIFT CIPHER(CAESAR CIPHER)" or not.

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

3. Used table showing letter frequencies in English language from the lecture slides and internet.

Observations:

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

2. We tried numerous shifts with Caesar Cipher until we concluded this wasn't the one because none of the 26 shifts made sense.

3. For Substitution Cipher, we first determined the frequencies of all the characters before moving on to the mappings.

4. From our initial analysis, 'C' was the most occurring letter which we substituted with 'e' which is the most occurring letter in English. Similarly 'F' was substituted with 't'. ‘Y’ has the same frequency as ‘d’ in the English language.

5. We concluded that 'I' will be equivalent to 'h' as we had a word 'fic' which maps to 't_e'.

6. The delimiters were supposed to be reserved in mono alphabetic substitution cipher, so, after looking at the text(Example - ‘c!Fi’) we could make out that there was a change in the number of spaces. Also, the word ‘Fi c’ left us with ‘Th e’ was a clear indication of mismatched spaces. After solving the issue created by spaces the text became more clearer. The word ‘fio kokfice’ should be ‘fiok ok fic e’ which is ‘this is the _’ as ‘OK’ and ‘thOK’ could only be ‘is’ therefore giving the mapping of ‘O’ to ‘i’ and ‘K’ to ‘s’.

7. ‘LithGVt’ could be decoded to only one word that is ‘without’. Giving us 'L' substituted by 'w' ,'G' substituted by 'o', and 'V' by 'u'. After these substitutions we got ‘Xuotes’ that

could be decoded to only one word that is ‘quotes’, therefore 'X' was replaced by 'q'.

8. ‘s uN s titutioQ’ looked very convincing as ‘substitution’. So we replaced 'N' with 'b' and 'Q' with 'n'. ‘wh iPh’ corresponded to ‘which’, thus replacing 'P' with 'c' and getting ‘ciJher’ as ‘cipher’ and making ‘pHsswor d’ as ‘password’.

9. ‘pUaces’ becomes ‘places’ and similarly we were getting letters decoded one after another with meaningful formations. After taking a lot of attempts we got the mapping of all the letters of the cipher text.

10. After all the cipher text was replaced by their substitute text, we observed that the first sentence had been rotated by 12 characters, left cyclic.

11. It is given in the plaintext that every digit has been shifted by 2 places. Since 2 is also a digit, let us assume that the initial digit in place of 2 was X which has been converted to 2 after shifting, so the conclusion that we drew is that every digit in cipher text has been shifted by X, thus needing to shift X by X to get 2. The equation for finding X becomes:

X + X = 2 mod 10

After solving this we got X as 1 or 6.

Applying the values in the password so obtained, we get the following two passwords:

1. iRqy3U5qdgt
2. iRqy8U0qdgt

We tried both and the first one was accepted.

Q4 Mapping

10 Points

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

Plaintext space and ciphertext space are the sets of strings composed of uppercase English alphabets, lowercase English alphabets, digits, spaces and punctuations.

CIPHERTEXT SPACE:

"omkf pi hdn cmgef icphsck .H krg vphqkc c, fic mco kqgf ioqag eo qfcmckf oq ficpihdn cm .Kg dcgeficu hfcm pi hdn cmklo uuncdgmc oqfc mc kfoq afihqfiokgq c!Fi cpgy cvkc yeg mfio kdck kha cokh kodjuck vn k fofvfo gqpojicmoqli opiyoa of kihsc nccqki oefc ynr2 juhpck. Fi c jhkklgm yok oMxr9V1x ya flofigvffic xvgfck. Fio kokfice"

The PLAINTEXT SPACE after substitution and rotation:

“this is the first chamber of the caves. As you can see, there is nothing of interest in the chamber. Some of the later chambers will be more interesting than this one. The code used for this message is a simple substitution cipher in which digits have been shifted by 2 places. The password is iRqy3U5qdgt without the quotes.”

The following mapping is extracted from the explanation in question 3:
A→g, C→e, D→m, E→f, F→t, G→o, H→a, I→ h, J→p, K→s, L→w, M→r, N→b, O→i, P→c, Q→n, R→y, S→v, U→l, V→u, X→q, Y→d, 2→6, 9→3, 1→5

Q5 Password
5 Points

What is the final command used to clear this level?

iRqy3U5qdgt

Q6 Codes
0 Points

Upload any code that you have used to solve this level

▾ tools.ipynb Download

In [12]:
```
import string
characters = string.ascii_uppercase
```

In the following piece of code we have removed the extra spaces so that it could be used for UNIGRAM and BIGRAM analysis

In [13]:
```
cipher_text = "omkf pi hdn cmgef
icphsck .H krg vphqkc c,fic mco
kqgf ioqag eo qfcmckf oq ficpihdn
cm .Kg dcgeficu hfcm pi hdn cmklo
uuncdgmc oqfc mc kfoq afihqfiokgq
c!Fi cpgy cvkc yeg mfio kdck kha
cokh kodjuck vn k fofvfo
gqpojicmoqli opiyoa of kihsc nccqki
oefc ynr2 juhpck. Fi c jhkklgm yok
oMxr9V1x ya flofigvffic xvgfck. Fio
kokfice"
cipher_text =
cipher_text.upper().replace(" ",
"")
print("Cipher text:\n",cipher_text)
```

```
Cipher text:
 OMKFPIHDNCMGEFICPHSCK.HKRGVPHQKCC,FICMC
```

In the following piece of code

In the following piece of code we wish to analyse the frequency of every UNIGRAM

In [14]:

```
frequencies =
{i:cipher_text.count(i)/len(cipher_text)
for i in characters}
sorted(frequencies.items(), key = lambda
x[1], reverse = True)
```

Out [14]:

```
[('C', 13.48314606741573),
 ('F', 10.486891385767791),
 ('K', 10.112359550561797),
 ('O', 9.363295880149813),
 ('I', 8.239700374531834),
 ('G', 5.2434456928838955),
 ('H', 4.868913857677903),
 ('M', 4.868913857677903),
 ('Q', 4.49438202247191),
 ('P', 3.3707865168539324),
 ('D', 2.6217228464419478),
 ('N', 2.6217228464419478),
 ('V', 2.6217228464419478),
 ('E', 2.247191011235955),
 ('Y', 2.247191011235955),
 ('A', 1.8726591760299627),
 ('U', 1.8726591760299627),
 ('J', 1.4981273408239701),
 ('L', 1.4981273408239701),
 ('R', 1.1235955056179776),
 ('X', 1.1235955056179776),
 ('S', 0.7490636704119851),
 ('B', 0.0),
 ('T', 0.0),
 ('W', 0.0),
 ('Z', 0.0)]
```

Frequency analysis of BIGRAMS

In [15]:

```
big_fr =
{i+j:cipher_text.count(i+j) for i
in characters for j in characters
if cipher_text.count(i+j)>0}
p =sum(big_fr.values())
for i in big_fr:
    big_fr[i] =
float(big_fr[i]/p)*100
sorted(big_fr.items(), key = lambda
x: x[1], reverse = True)
```

Out [15]:

```
[('FI', 5.64516129032258),
 ('IC', 3.629032258064516),
 ('CM', 3.225806451612903),
 ('CK', 2.82258064516129),
 ('OK', 2.82258064516129),
 ('IO', 2.4193548387096775),
 ('OQ', 2.4193548387096775),
 ('FC', 2.0161290322580645),
 ('IH', 2.0161290322580645),
 ('KF', 2.0161290322580645),
 ('NC', 2.0161290322580645),
 ('FO', 1.6129032258064515),
 ('MC', 1.6129032258064515),
 ('PI', 1.6129032258064515),
 ('QF', 1.6129032258064515),
 ('CO', 1.2096774193548387),
 ('CP', 1.2096774193548387),
 ('DN', 1.2096774193548387),
 ('EF', 1.2096774193548387),
 ('GE', 1.2096774193548387),
 ('GM', 1.2096774193548387),
 ('HD', 1.2096774193548387),
 ('HK', 1.2096774193548387),
 ('KO', 1.2096774193548387),
 ('OF', 1.2096774193548387),
 ('AF', 0.8064516129032258),
 ('CC', 0.8064516129032258),
 ('CY', 0.8064516129032258),
 ('DC', 0.8064516129032258),
 ('GF', 0.8064516129032258),
 ('GQ', 0.8064516129032258),
 ('GV', 0.8064516129032258),
 ('HQ', 0.8064516129032258),
 ('HS', 0.8064516129032258),
 ('JU', 0.8064516129032258),
 ('KC', 0.8064516129032258),
 ('KG', 0.8064516129032258),
 ('KH', 0.8064516129032258),
 ('KI', 0.8064516129032258),
 ('KK', 0.8064516129032258),
 ('KL', 0.8064516129032258),
 ('LO', 0.8064516129032258),
 ('MK', 0.8064516129032258),
 ('OM', 0.8064516129032258),
 ('PH', 0.8064516129032258),
 ('QA', 0.8064516129032258),
 ('QK', 0.8064516129032258),
 ('SC', 0.8064516129032258),
 ('UH', 0.8064516129032258),
 ('VF', 0.8064516129032258),
 ('YO', 0.8064516129032258),
 ('AC', 0.4032258064516129),
 ('AG', 0.4032258064516129),
 ('AO', 0.4032258064516129),
 ('CD', 0.4032258064516129),
 ('CE', 0.4032258064516129),
 ('CG', 0.4032258064516129),
 ('CJ', 0.4032258064516129),
 ('CN', 0.4032258064516129),
 ('CQ', 0.4032258064516129),
 ('CU', 0.4032258064516129),
 ('CV', 0.4032258064516129),
 ('CX', 0.4032258064516129),
 ('DG', 0.4032258064516129),
 ('DJ', 0.4032258064516129),
 ('EG', 0.4032258064516129),
 ('EO', 0.4032258064516129),
 ('FF', 0.4032258064516129),
 ('FK', 0.4032258064516129),
 ('FL', 0.4032258064516129),
 ('FP', 0.4032258064516129),
 ('FV', 0.4032258064516129),
 ('GD', 0.4032258064516129),
 ('GY', 0.4032258064516129),
 ('HA', 0.4032258064516129),
 ('HF', 0.4032258064516129),
 ('HP', 0.4032258064516129),
 ('IG', 0.4032258064516129),
 ('IY', 0.4032258064516129),
 ('JH', 0.4032258064516129),
 ('JI', 0.4032258064516129),
 ('KD', 0.4032258064516129),
 ('KQ', 0.4032258064516129),
 ('KR', 0.4032258064516129),
 ('KV', 0.4032258064516129),
 ('LG', 0.4032258064516129),
 ('LI', 0.4032258064516129),
 ('MF', 0.4032258064516129),
 ('MG', 0.4032258064516129),
 ('MO', 0.4032258064516129),
 ('MP', 0.4032258064516129),
 ('MX', 0.4032258064516129),
 ('MY', 0.4032258064516129),
 ('NK', 0.4032258064516129),
 ('NR', 0.4032258064516129),
 ('OA', 0.4032258064516129),
 ('OD', 0.4032258064516129),
 ('OE', 0.4032258064516129),
 ('OG', 0.4032258064516129),
 ('OJ', 0.4032258064516129),
 ('OP', 0.4032258064516129),
 ('OU', 0.4032258064516129),
 ('PC', 0.4032258064516129),
 ('PG', 0.4032258064516129),
 ('PO', 0.4032258064516129),
 ('QC', 0.4032258064516129),
 ('QG', 0.4032258064516129),
 ('QL', 0.4032258064516129),
 ('QP', 0.4032258064516129),
 ('RG', 0.4032258064516129),
 ('UC', 0.4032258064516129),
 ('UN', 0.4032258064516129),
 ('UU', 0.4032258064516129),
 ('VG', 0.4032258064516129),
 ('VK', 0.4032258064516129),
 ('VN', 0.4032258064516129),
 ('VP', 0.4032258064516129),
 ('XR', 0.4032258064516129),
 ('XV', 0.4032258064516129),
 ('XY', 0.4032258064516129),
 ('YA', 0.4032258064516129),
 ('YC', 0.4032258064516129),
 ('YE', 0.4032258064516129),
 ('YN', 0.4032258064516129)]
```

Substitution Cipher using Frequency Analysis

In [16]:

```
plain_text = ""
for ch in cipher_text:
    if ch=='C':
        ch='E'
    elif ch=='F':
        ch='T'
    elif ch=='K':
        ch='S'
    elif ch=='O':
        ch='I'
    elif ch=='I':
        ch='H'
    elif ch=='G':
        ch='O'
    elif ch=='H':
        ch='A'
    elif ch=='M':
        ch='R'
    elif ch=='Q':
        ch='N'
    elif ch=='P':
        ch='C'
    elif ch=='D':
        ch='M'
    elif ch=='N':
        ch='B'
    elif ch=='V':
        ch='U'
    elif ch=='E':
        ch='F'
    elif ch=='Y':
        ch='D'
    elif ch=='A':
        ch='G'
    elif ch=='U':
        ch='L'
    elif ch=='J':
        ch='P'
    elif ch=='L':
        ch='W'
    elif ch=='X':
        ch='Q'
    elif ch=='S':
        ch='V'
    elif ch=='R':
        ch='Y'
    plain_text+=ch
print("Deciphered text:")
print(plain_text)
```

```
Deciphered text:
IRSTCHAMBEROFTHECAVES.ASYOUCANSEE,THEREI
```

Q7 Team Name

0 Points

Enigma

Assignment 1

● GRADED

GROUP

Pranshu Sahijwani
Gajender Sharma
Kajal Sethi
View or edit group

TOTAL POINTS

48 / 50 pts

QUESTION 1	
Commands	R **5** / 5 pts
QUESTION 2	
Cryptosystem	**5** / 5 pts
QUESTION 3	
Analysis	**25** / 25 pts
QUESTION 4	
Mapping	**8** / 10 pts

QUESTION 5

Password **5** / 5 pts

QUESTION 6

Codes **0** / 0 pts

QUESTION 7

Team Name **0** / 0 pts