## **Q1** Team Name
0 Points

ANV

## **Q2** Commands
10 Points

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

$\textbf{Commands used} : -$
$\bullet\, To\; obtain\; the\; hints : \;\textbf{go}$
$\bullet\, To\; go\; back\; from\; hints : \;\textbf{back}$
$\bullet\, To\; reach\; the\; cipher\; text : \;\textbf{read}$

$\textbf{After executing the above commands in sequence, we reached the cip}$

## **Q3** CryptoSystem
10 Points

What cryptosystem was used in this level?

• **”PLAYFAIR CIPHER”** is used for Encryption and Decryption of text.

• **KEY**: – **”CRYPTANALYSIS”** to Encrypt and Decrypt the text.

## **Q4** Analysis

20 Points

What tools and observations were used to figure out the cryptosystem? (Explain in less than 300 words)

■**OBSERVATIONS**: –

**i.** when we first saw the **Ciphertext**, we assumed it to be a substitution cipher, and a simple substitution cipher with frequency analysis **failed** to decrypt the message.

**ii.** After entering **”GO”** command, there is a **hint** provided in the last line which says **”PLAYFAIR”**., this lead to the thought of **PLAYFAIR CIPHER.**

**iii.** A key observation in Ciphertext is
**no two letters in a digraph are the same**
(a possible case would be "XX" present in plain text but not available in the given ciphertext)
and letter **”j”** is absent entirely, which confirmed that PLAY FAIR is used for the encryption.

**iv. KEY** for playcipher is also provided as a **hint**, but in **”MORSECODE”**, which we decrypted and found out to be **”CRYPTANALYSIS”.**

**v.** To confirm and check the correctness of the above key, we **automated** the decryption process of the playcipher algorithm with multiple possible keys (i.e. words) taken from the file **"words.txt".**

(Our intuition was that PLAYFAIR CIPHER might have English vocabulary as its KEY)

"words.txt" consists of **416296** different words of varying lengths that are possible to be a key in Playfair cipher.

**vi.** DECRYPTION program also gave us **"CRYPTANALYSIS"** as the key.

■**TOOLS**: –

**i. Automated program** to find the KEY among the possible vocabulary words in **"words.txt"**.

**ii.** Using the **KEY**, we decrypted the Ciphertext by **PLAYFAIR cipher**.

**iii. "words.txt"** file is used to automatically find the key.

**RESOURCE links :**

1.) http://practicalcryptography.com/media/cryptanalysis/files/english_quadgrams.txt.zip
2.) https://github.com/dwyl/english-words/blob/master/words.txt

## **Q5** Decryption Algorithm

15 Points

Briefly describe the decryption algorithm used. Also mention the plaintext you deciphered. ( Use less than 350 words)

■**DECRYPTION ALGORITHM** => **"PLAYFAIR CIPHER"**

**Playfair cipher** is a *"digram"* substitution cipher
which follows the manual symmetric encryption technique.

●The Playfair cipher uses a **5x5** matrix in which unique letters of the keyword
are filled first left to right, and subsequently, all the remaining
unique alphabets are filled in alphabetical order.
(note:- LETTER **"I"** and **"j"** will occupy only one cell of the matrix, which enables to fit 26 alphabets in a 5*5 matrix).

●Next, form the Digrams of the ciphertext, we decrypt each digram using the matrix.

●**RULES FOR DECRYPTION** : –

**1**)If both the letters belong to the **same row**, then take the letter to the left of each one from the matrix in a WRAP-AROUND manner, respectively.

**2**)If both the letters belong to the **same column**, then take the letter above to each one from the matrix in a WRAP-AROUND manner, respectively.

**3**)If **neither** of the above rules is true, then form a quadrilateral with the two letters and take the letters on the other opposite diagonal respectively.

## ■<u>**ALGORITHM**</u>:-

1)**Score()** function calculates the sum of Quadgram scores of ciphertext.

2)**rules_func()** performs decryption using  above RULES FOR DECRYPTION.

3)**matrix_formation()** builds a 5*5 matrix based on the KEY given.

4)Firstly, try to find the **KEY** among all possible keys from
the file **"words.txt"**.

5)Iterating through each *<u>word</u>* (possible key),perform **matrix_formation()** and calculate score using **Score()** function.

6)pick the **OPTIMAL** word of the high score, which serves as the **"KEYWORD"** for our analysis.

7)Decrypt the ciphertext using KEYWORD, by performing matrix_formation(), which creates the matrix and then perform decryption digram-wise using rules_func().

8)we get the **decryptedPlaintext**.

**End**

## ■<u>**MANUAL CORRECTIONS**</u> :-

**i.** As **"X"** will be attached for repeating letters during encryption

process ,these are corrected manually.
[**"OUTX"** => **"OUT"**,
,**"WILXL"** => **"WILL"**,
**"NEXED"** => **"NEED"**].

**ii. "IOY"** => **"JOY"** , as **"J"** is substituted as **"I"** in PLAYFAIR CIPHER.

■**PLAINTEXT** => "BE WARY OF THE NEXT CHAMBER, THERE IS VERY LITTLE JOY THERE. SPEAK OUT THE PASSWORD "ABRA_CA_DABRA" TO GO THROUGH. MAY YOU HAVE THE STRENGTH FOR THE NEXT CHAMBER. TO FIND THE EXIT YOU FIRST WILL NEED TO UTTER MAGIC WORDS THERE."

## **Q6** Password

10 Points

What was the final command used to clear this level?

ABRA_CA_DABRA

## **Q7** Code

0 Points

Upload any code that you have used to solve this level

▾ assgn2.ipynb Download

In [1]:

```
from math import log
```

In [2]:

```
#DECRYPTION FUNCTION BASED ON THE MATRIX
def rules_func(a,b):

    r1,c1 = ind[a]
    r2,c2 = ind[b]

#     #new indices after decryption
#     x1=y1=0    #for 1st letter of digram
#     x2=y2=0  #for 2nd letter of digram

    if r1 == r2:     #BOTH LETTERS IN SAME ROW OF MATRIX
        x1 = x2 =r1
        y1 = (c1-1)%5
        y2 = (c2-1)%5
    elif c1 == c2:     #BOTH LETTERS IN SAME COLUMN OF MATRIX
        y1 = y2 = c1
        x1 = (r1-1)%5
        x2 = (r2-1)%5
    else:
        x1 = r1
        y1 = c2
        x2 = r2
        y2 = c1
    return matrix[x1][y1]+matrix[x2][y2]
```

In [3]:

```
#SCORE FUNCTION
def score():
    sc =0                #score
    text = ""            #DECRYPTED TEXT

    for i in range(0,l,2):    #DECRYPTING each bigram
        text+= rules_func(cipher[i],cipher[i+1])

    for i in range(len(text)-3):        #as last three letters cannot
form quadgram
        quad_gram = text[i:i+4]

        if quad_gram in quad:
            sc+=log(quad[quad_gram],10)
    return sc
```

In [4]:

```
#MATRIX FILLING
def matrix_formation(w):
    i=j=0
    alp=set('ABCDEFGHIKLMNOPQRSTUVWXYZ')

    #FILLING MATRIX BASED ON WORD(i.e possible key)
    for k in range(len(w)):
        if w[k] in alp:
            alp.remove(w[k])
            matrix[i][j] = w[k]

            if j==4:        #END OF ROW
                j=0
                i+=1
            else: j+=1

    #FILLING REMAINING ALPHABETS INTO MATRIX
    for k in sorted(alp):
        matrix[i][j] = k

        if j==4:        #END OF ROW
            j=0
            i+=1
        else:
            j+=1

    #ASSIGNING INDEX TO EACH ALPHABET OF THE MATRIX
    for i in range(5):
        for j in range(5):
            ind[matrix[i][j]]=(i,j)
```

In [5]:

```
#CREATING A DICTIONARY FOR QUADGRAMS
quad ={}

#READING FILE
file = open("english_quadgrams.txt")
for i in file:
    a,b = i.split()
    quad[a] = int(b)
```

```
#LENGTH OF THE QUADGRAM DICTIONARY
n = len(quad)
```

In [6]:

```
#PREPROCESSING "words.txt file"
words=[]
for i in open("words.txt"):
    i=i.upper().replace("\n",'')         #REPLACING NEWLINE CHAR

    #DEALING WITH ONLY ENGLISH_DICTIONARY WORDS
    flag=True
    for j in i:
        if not (ord(j)>=65 and ord(j)<=90):
            flag=False
            break
    if flag:
        words.append(i)

len(words)
```

Out [6]:

```
416296
```

In [7]:

```
#CIPHER TEXT
cipher_text="DF ULYP XO CQD LFWC RUBHEDY, CQDYG LN XDYL EGIYIG LMP
CQDYF. LYFNH HXPZ CQF YNILXKPB \"NDCB_AN_BBHCN\" PQ FQ CQPKZBK. OLC
PMC UNUG YMB IPYDIDCQ OXY CMB LDZP AULHDFY. CX OALG RMB FWGI PMX
BNTIP ZLSWS LFWFE PQ ZCYGY KIBAT XMNKI PMBYD."
cipher=''

#DEALING WITH ONLY ALPHABETS
for i in cipher_text:
    if ord(i)>=65 and ord(i)<=90:
        cipher+=i
l=len(cipher)
```

In [8]:

```
ind={}    #STORES INDEX OF EACH ALPHABET IN THE MATRIX
sc=0      #SCORE
matrix=[['' for i in range(5)] for j in range(5)]          #5*5

MATRIX
keyword=''
```

## FINDING KEY FOR PLAYFAIR CIPHER ANALYSIS

In [9]:

```
for w in words:                 #CHECKING FOR EACH WORD POSSIBILTY OF
BEING A KEY
    matrix_formation(w)

    new_sc=score()
#     print('newsc is: ',new_sc)
    if new_sc>sc:
        sc=new_sc
        keyword=w

print("KEY IS: = > ",keyword)
```

```
KEY IS: = >  CRYPTANALYSIS
```

## DECRYPTING WITH THE KEY "CRYPTANALYSIS"

In [10]:

```
# CREATING MATRIX WITH THE KEY
matrix_formation(keyword)
```

In [11]:

```
#DECRYTING THE CIPHER TEXT WITH "KEYWORD"
r=''
for i in range(0,l,2):
        r+=rules_func(cipher[i],cipher[i+1])
```

In [12]:

```
plain_text=''
j=0

for i in range(len(cipher_text)):
    if ord(cipher_text[i])>=65 and ord(cipher_text[i])<=90:
            plain_text+=r[j]
            j+=1
        else:
            plain_text+=cipher_text[i]
print(plain_text)
```

```
BE WARY OF THE NEXT CHAMBER, THERE IS VERY LITTLE IOY THERE. SPEAK OUTX THE
```

In [13]:

```
#MANUAL CORRECTIONS
plain_text = plain_text.replace("OUTX","OUT")
plain_text = plain_text.replace("WILXL","WILL")
plain_text = plain_text.replace("NEXED","NEED")

plain_text = plain_text.replace("IOY","JOY")
```

In [14]:

```
plain_text
```

Out [14]:

```
'BE WARY OF THE NEXT CHAMBER, THERE IS VERY LITTLE JOY THERE. SPEAK OUT THE
```

In [ ]:

words.txt Download

```
1  Large file hidden. You can download it using the button above.
```

english_quadgrams.txt Download

```
1  Large file hidden. You can download it using the button above.
```

## Assignment 2

● GRADED

GROUP
Dibbu Amar Raja
Vikas
Idamakanti Venkata Nagarjun Reddy
View or edit group

TOTAL POINTS
**65 / 65 pts**

| | |
|---|---|
| QUESTION 1 | |
| Team Name | **0** / 0 pts |
| QUESTION 2 | |
| Commands | **10** / 10 pts |
| QUESTION 3 | |
| CryptoSystem | **10** / 10 pts |
| QUESTION 4 | |
| Analysis | **20** / 20 pts |
| QUESTION 5 | |
| Decryption Algorithm | **15** / 15 pts |
| QUESTION 6 | |
| Password | **10** / 10 pts |

QUESTION 7

Code

**0** / 0 pts