Q1 Commands
5 Points

List the commands was used in this level?

main chamber..
go -this is a small chamber.....
put- someone bit hands
back
Enter-mushrooms growing.
pluck-you plucked mashrooms
back
give- trapped spirit
back
back- to the main chamber
thrnxxtzy- glass panel
read-ciphertext
the_magic_of_wand

Q2 Cryptosystem
10 Points

What cryptosystem was used in the game to reach the password?

Substitution-permutation block cipher popularly known as SP cipher
block length=5.

$Mono-alphabetic$ substitution of cipher text, then a particular permutation of the block size=5. The permutation is done after removing spaces, underscore, and commas. Basically, anything other than characters is removed. It is a $single$ permutation used in all blocks except the last block(because the length of the last block is 4, which is less than the block size, i.e. 5). decryption permutation key is 43512. Encryption and decryption keys are inverses of each other

keys are inverses of each other.

Q3 Analysis
30 Points

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

Observing Ciphertext

Started with frequency analysis, bigram frequency, trigrams, of cipher text. from the frequency analysis, we got 'q' as the most frequent character. 'v' as the next most frequent and so on. The frequency of a single character is listed in descending order. Since frequencies are not flattened, this hints that there may be substitutions.

Q	30×	10.56%
V	29×	10.21%
A	23×	8.1%
C	22×	7.75%
W	19×	6.69%
F	19×	6.69%
L	17×	5.9%
T	13×	4.58%
Y	13×	4.58%
S	11×	3.87%
P	11×	3.87%
N	10×	3.52%
R	9×	3.17%
X	8×	2.82%
G	8×	2.82%
E	7×	2.46%
D	7×	2.46%
J	6×	2.11%
U	6×	2.11%
M	5×	1.76%

H 5× 1.76%
B 3× 1.06%
K 2× 0.7%
I 1× 0.35%

known plaintext attack

The cipher text contains a word 'nqg_vfusr_ec_wawy'. From our observation of the previous levels we guessed that an 8 letter word near 'nqg_vfusr_ec_wawy' can be 'password'. Our observation helps us do the known plaintext attack.

the password —> vml lhvqpawr

Decryption

We start our decryption process by checking some of the probable methods.

>*checking ceaser's and substitution cipher*
checking caesar's cipher and substitution by noticing that we have 'ss' in the plaintext(password) but no corresponding repeated words in ciphertext(lhvqpawr).

>*checking permutation cipher*
we do not find shuffling of 'the password' meaning it is not the permutation only. Also, frequency analysis suggests that there may be mono-alphabetic substitution.

>*checking vigenere cipher*
We got code like word $'thrnxxtzy'$. we tried vigenere with $'thrnxxtyz'$ and 'jcjcffcccb' (from the previous level) . Result from vigenere with key 'thrnxxtzy' are

"xfwwyvd uw ylpl sofw cqtq ms gyysswn
mo gwd ytpmfov gzoqyj ocjrlysd ij haa ooiz. iy dhsfi,
ydt stjt a dlz.....
This does not make any sense.

>$SPcipher$

Finally, we tried substitution-permutation cipher. It is performed by breaking the text into the block. From the ciphertext, we removed spaces, underscore, commas, basically, anything other than character is removed. We tried with block different block lengths. Because 2, and 3 are small block sizes and do not provide much security so directly tried block size 4.

we broke the cipher text into 4 lengths and tried a known-plaintext attack. * means not-known.

"qmnj|vsan|vwew|cflc|tvpr|jtjt|vvpl|vlfv|xjav|qild|hcx
m|lnvc|

nacy|clpa|fcgy|tvfv|wfvw|gqyp|pqqp|qcsy|wsqr|xqm
n|jvaf|
  ycgv|tlvh|fcwt|ylae|uqfv ....
vmll|hvqp|awrn|
qgvf|usre|cwaw|yqpf|nwga|wdgf|"

$$\text{vmll hvqp awrn} -> \text{thep assw ord*}$$

we do not get 2 same characters in 'hvqp'
corresponding to 'assw'. we let us try with block size 5.

"qmnjv|sanvw|ewcfl|ctvpr|jtjtv|vplvl|fvxja|vqild|hcxm
l|nvcna|

cyclp|afcg|tvfvw|fvwgq|yppqq|pqcsy|wsqrx|qmnjv|af
ycg|vtlvh|

fcwty|laeuq|fvxja|tkbvc|qnsqs|lhfav|awncc|veasf|uqb
qv|qtcyl|

lrqrx|xwacf|ypsdc|uqfav|rqcge|fqpya|ttrac|xwvta|aw
wdd|veasf|

lcbqv|dtraw|mvupq|quwxd|ecgqc|wtyqy|aflvl|qsyqk|lhqsn|afqvm|
llhvq|pawrn|qgvfu|srecw|awyqp|fnwga|wdgf"

$$\text{afqvm llhvq pawrn} -> ***\text{th epass word}*$$

This block size is probably correct because two 's' corresponds to two 'l'. So we tried all possible 5!. By noticing the position 'ss' permutation key end with 12 or 21. So now we have to try only 3!*2=12 permutations only.

$$\text{llhvq} -> \text{epass}$$

The frequency analysis shows that 'v' and 'q' are very frequent characters. This implies that probably 'v' and 'q' will not be p. so 'h' can be replaced by 'p'. so we guessed 3 out of 5 characters. So we try all 2!*2= 4 permutations.

['l'->'s'
'h'->'p']

53412 43512 53412 43512

we found all the permutations with the help of simple c++ code. After trying with all 4 permutations along with other texts. checked some more texts and finally came out with the permutation key as 43512.

In the cipher block 'llhvq' we guessed l,h. We are left with 'v' and 'q'. One of them is 'e' and other is 'a'. By looking at the cipher text we observe that the character 'q' is always near the single character word. So 'q' can be 'a' and 'v' can be 'e'.

['q'->'a'

[q ->a
'v' ->'e']

We replaced the found subtitution and Re-arranged the cipher text according to the permutation key that helped in further maping of substitutions.

Demo:
..... afqvm llhvq pawrn.... $after\ permuation$
.....vqmaf vhqll rwnpa.....

we also know that it corresponds to ***th epass word*

['a'->'t'
'f' -> 'h'
'r'->'w'
'w'->'o'
'n'->'r'
'p'->'d']

>**subtitution mappings**
cipher text -> plaintext

' ' -> ' ' (spaces)
' . ' -> '.' (full stop)
'_' -> '_' (underscore)
' , ' -> ' ,' (comma)
'q'->'a',
'j'->'b',
'e'->'c',
'p'->'d,
'v'->'e',
's'->'f',
'g'->'g',
'f'->'h',
'c'->'i',
'm'->'k',
't'->'l',

'u'-> 'm',
'y'->'n',
'w'->'o',
'h'->'p',
'i'->'q',
'n'->'r',
'l'->'s',
'a'->'t',
'd'->'u',
'b'->'v',
'r'->'w',
'x'->'y',
'k'->'j'

last block

One more thing to be noticed is that the last block is of 4 lengths only so permutation is not performed in the last block. only substitution is performed in the last block.

how 'k'->'j'

we got substitutions of all characters with the help of frequency analysis and word guessing. But one character 'k' was left. jaffar. This word did not appear anywhere in the previous level. We recalled about a Disney character named jaffar-the evil magician so we mapped it as 'k'->'j' .

plaintext

"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 caves. it would make you a magician, no less than jaffar! speak the password the_magic_of_wand to go through"

Q4 Password
5 Points

What was the final command used to clear this level?

the_magic_of_wand

Q5 Codes
0 Points

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

Breaking_into_blocks.cpp Download

```cpp
#include <iostream>

using namespace std;

int main()
{

    string str="qmnjvsanvwewcflctvprjtjtvvplvlfvxjavqildhcxr


    // string str="breakerofthiscodewillbeblessedbythesqueakys
    affarthespiritofthecavemanisalwayswithyoufindther
    affartogothroughspeakthepassword"
   ; int j=0;

    for(int i=0; i<str.length();i++){
        j++;
        cout<<str[i];
        if(j==5){
            cout<<" ";
            j=0;

        }
    }

    return 0;
```

```
25 }
```

permutation_code.cpp

Download

```cpp
#include <bits/stdc++.h>
using namespace std;


void permutations(string& a, int l, int r)
{

    if (l == r)
        cout << a << endl;
    else {
        // Permutations made
        for (int i = l; i <= r; i++) {

            // Swapping done
            swap(a[l], a[i]);

            // Recursion called
            permute(a, l + 1, r);

            // backtrack
            swap(a[l], a[i]);
        }
    }
}

int main()
{
    string str = "ABC";
    int n = str.size();

    // Function call
    permutations(str, 0, n - 1);
    return 0;
}

```

SP_Cipher_.ipynb

Download

PROGRAM FOR SP BLOCK

CIPHER

In [64]:
```
def sp_block_cipher_decipher(cipherte
permutation_key, substitution_key):
    # Convert the permutation key to
of integers
    perm = [int(x) for x in permutati

    # Split the ciphertext into block
size 4
    blocks = [ciphertext[i:i+5] for i
range(0, len(ciphertext), 5)]

    # Inverse the permutation on each
    permuted_blocks = []
    for block in blocks:
        permuted_block = [None] * 5
        for i in range(5):
            permuted_block[perm[i]] =
block[i]

permuted_blocks.append(''.join(permut


    # Decipher each block using the
substitution key
    deciphered_blocks = []
    for block in permuted_blocks:
        deciphered_block = ''
        for c in block:
            deciphered_block +=
substitution_key[c]

deciphered_blocks.append(deciphered_b

    # Join the deciphered blocks into
single string
    plaintext = ''.join(deciphered_bl

    return plaintext
```

In [65]:
```
cipher = "qmnjvsa nv wewc flct
ypri ti tyyplyl fy xia ygildhc
```

```
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."
print("THE CIPHER TEXT IS \n")
print(cipher)
clean_cipher = "".join([c for c
in cipher if c.isalpha()])
print("\nTHE CLEAN CIPHER IS
\n")
print(clean_cipher)
print("")
print("")
print("The length of the cipher
is "+str(len(clean_cipher)))
print("")
print("")
```

```
THE CIPHER TEXT IS

qmnjvsa nv wewc flct vprj tj tvvplvl

THE CLEAN CIPHER IS

qmnjvsanvwewcflctvprjtjtvvplvlfvxjavq


The length of the cipher is 284
```

In [66]:

```
ciphertext = clean_cipher[:-4]
permutation_key = '34102'
substitution_key = {
        'q':'a',
        'j':'b',
```

```
        'e':'c',
        'p':'d',
        'v':'e',
        's':'f',
        'g':'g',
        'f':'h',
        'c':'i',
        'k':'j',
        'm':'k',
        't':'l',
        'u':'m',
        'y':'n',
        'w':'o',
        'h':'p',
        'i':'q',
        'n':'r',
        'l':'s',
        'a':'t',
        'd':'u',
        'b':'v',
        'r':'w',
        'x':'y',
        '?':'z'}
plaintext =
sp_block_cipher_decipher(ciphertext,
permutation_key, substitution_key)
plaintext += "ough"
print("")
print("THE DECIPHERED CLEAN
PLAINTEXT IS : \n")
print(plaintext)  # Output: ABCDEFG
```

```
THE DECIPHERED CLEAN PLAINTEXT IS :

breakerofthiscodewillbeblessedbythesc
```

In [67]:

```
k = ""
l = 0
m = [" ",".",",","!",":","_"]
list_of_words = []
for i in cipher:

    if (i not in m):
```

```
            if (l < len(plaintext)):
                k += plaintext[l]
            l += 1
        else:
            list_of_words.append(k)
            k = " "
        if (i in m):
            list_of_words.append(i)
    print("\n")
    print("THE DECIPHERED PLAINTEXT
    IS : \n")
    print("".join(list_of_words))
```

```
THE DECIPHERED PLAINTEXT IS :

breaker  of  this  code  will  be  bl
```

In [ ]:

frequency_analysis.py Download

```
1  import collections
2
3  def frequency_analysis(text):
4      # Remove any non-alphabetic characters and
   convert to lowercase
5      text = ''.join(c for c in text if
   c.isalpha()).lower()
6
7      # Count the frequency of each letter in the
   text
8      freq = collections.Counter(text)
9
10     # Sort the letters by frequency (most
   frequent first)
11     sorted_freq = sorted(freq.items(),
   key=lambda x: x[1], reverse=True)
12
13     # Print the results
14     for letter, count in sorted_freq:
```

```
15          print(letter, count)
16
17  text =
    "qmnjvsanvwewcflctvprjtjtvvplvlfvxjavqildhcxmlnv
18  frequency_analysis(text)
```

Q6 Group name
0 Points

the_cryptonics

Assignment 3

● Graded

Group

TANEYA SONI
ABHISHEK KUMAR PATHAK
SONAM
View or edit group

Total Points

50 / 50 pts

Question 1

Commands **5** / 5 pts

Question 2

Cryptosystem **10** / 10 pts

Question 3

Analysis R **30** / 30 pts

Question 4

Password **5** / 5 pts

Question 5

Codes **0** / 0 pts

Question 6

Group name **0** / 0 pts