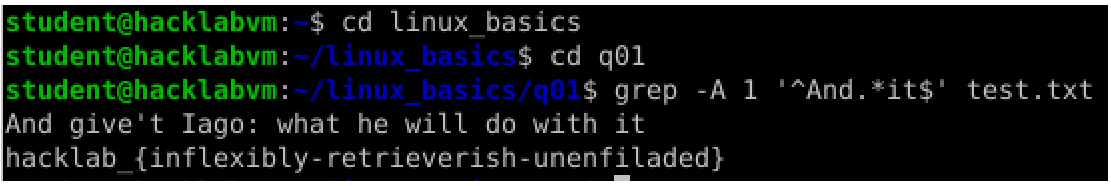
Part 1

Question 1:

The passphrase is hacklab_{inflexibly-retrieverish-unenfiladed}.

Screenshot:



```
student@hacklabvm:~$ cd linux_basics
student@hacklabvm:~/linux_basics$ cd q01
student@hacklabvm:~/linux_basics/q01$ grep -A 1 '^And.*it$' test.txt
And give't Iago: what he will do with it
hacklab_{inflexibly-retrieverish-unenfiladed}
```

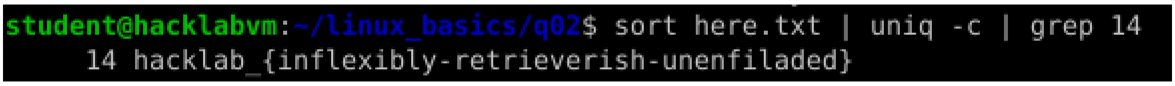
Explanation:

Grep is a command line tool for finding text quickly in files. I went to the q01 directory and used -A 1 to find the following the line that begins with the word "And" and ends with "it". "^" is the symbol for beginning of the line. "$" is the symbol for ending of the line. ".*" to match zero or more occurrences.

Question 2:

The passphrase that occurs 14 times is hacklab_{inflexibly-retrieverish-unenfiladed}.

Screenshot:



```
student@hacklabvm:~/linux_basics/q02$ sort here.txt | uniq -c | grep 14
     14 hacklab_{inflexibly-retrieverish-unenfiladed}
```

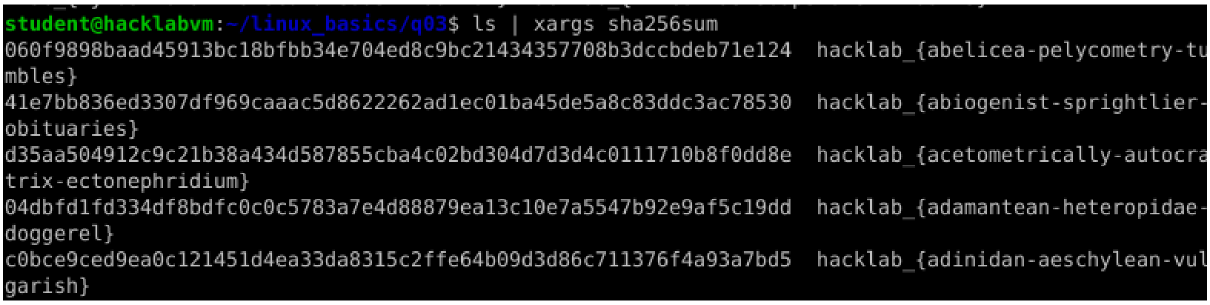
Explanation:

Sort the lines of here file. Then uniq -c will count the word occurrences. "|" is called pipe, it allows the use of multiple commands. Grep 14 to find the line that occurred exactly 14 times.

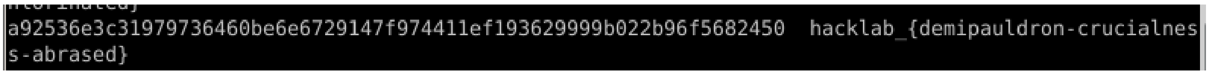
Question 3:

The name of the file with the SHA256 sum is hacklab_{demipauldron-crucialness-abrased}.

Screenshot:



```
student@hacklabvm:~/linux_basics/q03$ ls | xargs sha256sum
060f9898baad45913bc18bfbb34e704ed8c9bc21434357708b3dccbdeb71e124  hacklab_{abelicea-pelycometry-tu
mbles}
41e7bb836ed3307df969caaac5d8622262ad1ec01ba45de5a8c83ddc3ac78530  hacklab_{abiogenist-sprightlier-
obituaries}
d35aa504912c9c21b38a434d587855cba4c02bd304d7d3d4c0111710b8f0dd8e  hacklab_{acetometrically-autocra
trix-ectonephridium}
04dbfd1fd334df8bdfc0c0c5783a7e4d88879ea13c10e7a5547b92e9af5c19dd  hacklab_{adamantean-heteropidae-
doggerel}
c0bce9ced9ea0c121451d4ea33da8315c2ffe64b09d3d86c711376f4a93a7bd5  hacklab_{adinidan-aeschylean-vul
garish}
```



```
a92536e3c31979736460be6e6729147f974411ef193629999b022b96f5682450  hacklab_{demipauldron-crucialnes
s-abrased}
```

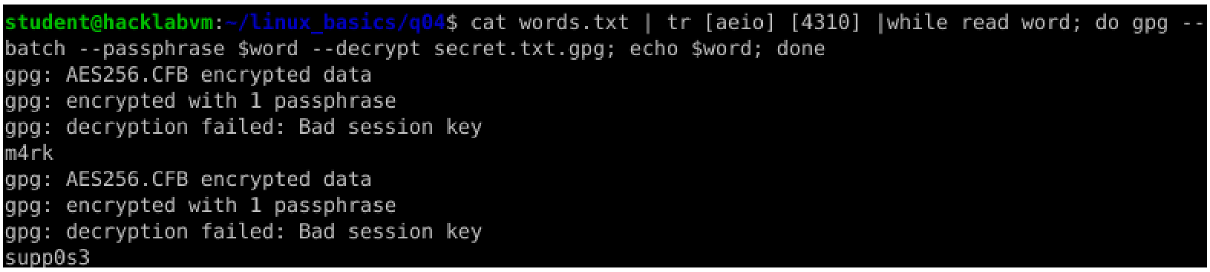
Explanation:

I was having trouble with grep "hash_value" so I had used the ls and xargs method. Xargs means extended arguments in which I searched sha256sum and scrolled through all the files with their sum till I found the correct sum.

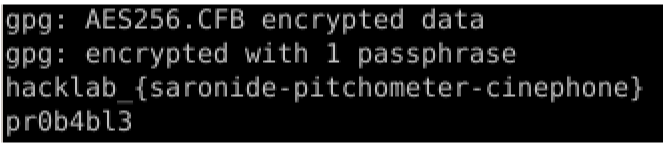
Question 4:

The correct password is pr0b4bl3 and the passphrase is hacklab_ {saronide-pitchometer-cinephone}.

Screenshot:



```
student@hacklabvm:~/linux_basics/q04$ cat words.txt | tr [aeio] [4310] |while read word; do gpg --
batch --passphrase $word --decrypt secret.txt.gpg; echo $word; done
gpg: AES256.CFB encrypted data
gpg: encrypted with 1 passphrase
gpg: decryption failed: Bad session key
m4rk
gpg: AES256.CFB encrypted data
gpg: encrypted with 1 passphrase
gpg: decryption failed: Bad session key
supp0s3
```



```
gpg: AES256.CFB encrypted data
gpg: encrypted with 1 passphrase
hacklab_{saronide-pitchometer-cinephone}
pr0b4bl3
```

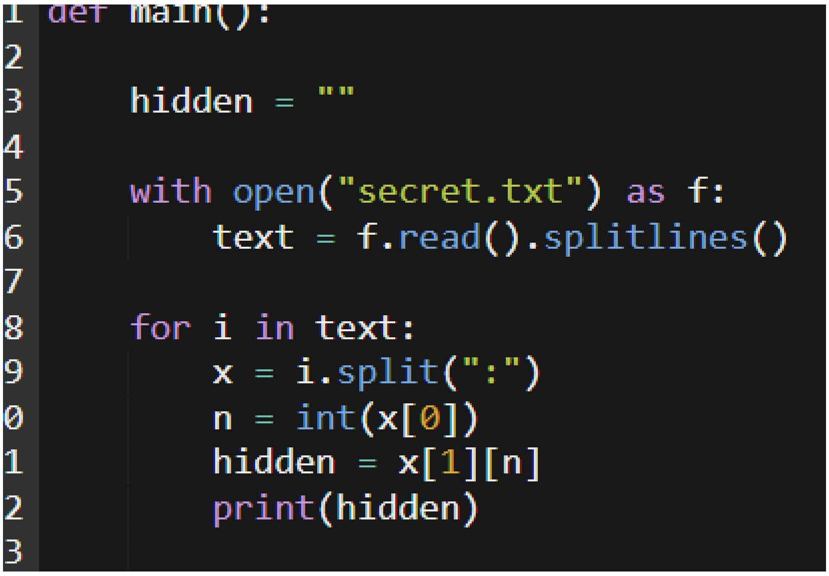
Explanation:

Using cat because we want to return the content of a file. Tr will translate to generate the correct passwords and the while loop command is provided in the question.

Question 5:

The flag hidden is hacklab_{hightailing-cothurnian-longhaired}.

Screenshot:



```
def main():

    hidden = ""

    with open("secret.txt") as f:
        text = f.read().splitlines()

    for i in text:
        x = i.split(":")
        n = int(x[0])
        hidden = x[1][n]
        print(hidden)
```

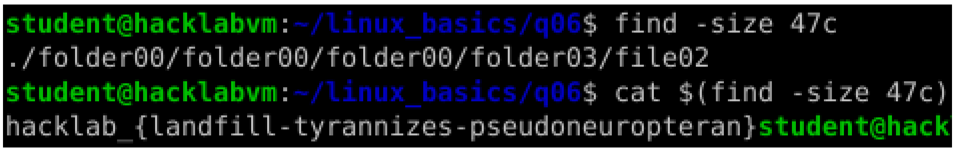
Explanation:

I used python program to decode the cipher with cyber.py. According to the cyber.py, the length of the random string appended before each character of the encoded text is indicated by the number at the beginning of each line.

Question 6:

The file directory is /folder00/folder00/folder00/folder03/file02 and the secret in it is hacklab_{landfill-tyrannizes-pseudoneuropteran}.

Screenshot:



```
student@hacklabvm:~/linux_basics/q06$ find -size 47c
./folder00/folder00/folder00/folder03/file02
student@hacklabvm:~/linux_basics/q06$ cat $(find -size 47c)
hacklab_{landfill-tyrannizes-pseudoneuropteran}student@hack
```

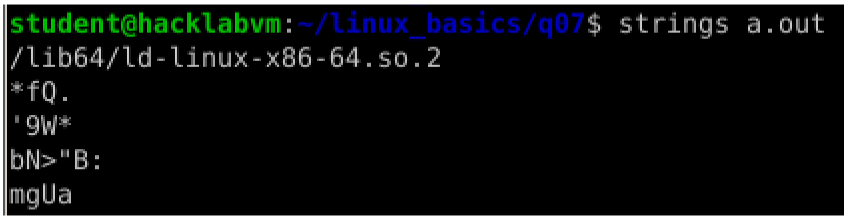
Explanation:

I first found the file path with -size 47c command. The second line was for the secret. The line just means find size of 47 bytes, -size 47c. The cat command allows for the secret to be shown.

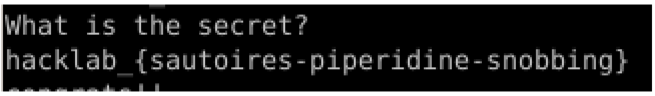
Question 7:

The secret is hacklab_{sautoires-piperidine-snobbing}.

Screenshot:



```
student@hacklabvm:~/linux_basics/q07$ strings a.out
/lib64/ld-linux-x86-64.so.2
*fQ.
'9W*
bN>"B:
mgUa
```



```
What is the secret?
hacklab_{sautoires-piperidine-snobbing}
```

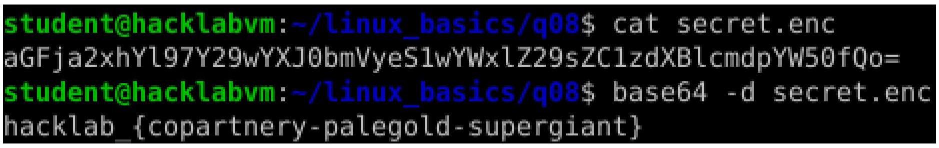
Explanation:

I executed a.out as mentioned in the question and tried to find the secret based on the output. I used strings because the secret is in a string format. So, I prompted all strings in the file and found the secret.

Question 8:

The secret is hacklab_{copartnery-palegold-supergiant}.

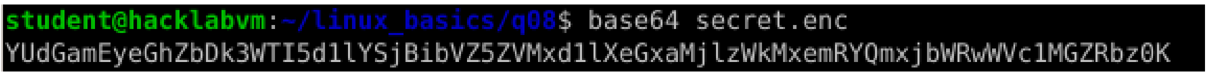
Screenshot:



```
student@hacklabvm:~/linux_basics/q08$ cat secret.enc
aGFja2xhYl97Y29wYXJ0bmVyeS1wYWxlZ29sZC1zdXBlcmdpYW50fQo=
student@hacklabvm:~/linux_basics/q08$ base64 -d secret.enc
hacklab_{copartnery-palegold-supergiant}
```

Explanation:

I first check the content using the cat command. Then I used base64 to decode for the secret. The -d is needed for the secret or else you will get code.



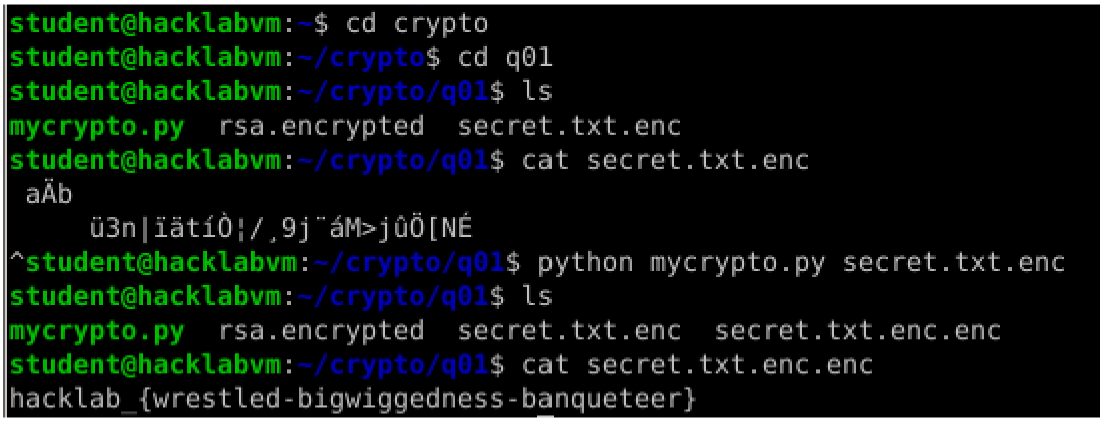
```
student@hacklabvm:~/linux_basics/q08$ base64 secret.enc
YUdGamEyeGhZbDk3WTI5d1lYSjBibVZ5ZVMxd1lXeGxaMjlzWkMxemRYQmxjbWRwWVc1MGZRbz0K
```

Part 2

Question 1:

The secret is hacklab_{wrestled-bigwiggedness-banqueteer}.

Screenshot:



```
student@hacklabvm:~$ cd crypto
student@hacklabvm:~/crypto$ cd q01
student@hacklabvm:~/crypto/q01$ ls
mycrypto.py  rsa.encrypted  secret.txt.enc
student@hacklabvm:~/crypto/q01$ cat secret.txt.enc
 aÄb
     ü3n|iätíÒ¦/¸9j¨áM>jûÖ[NÉ
^student@hacklabvm:~/crypto/q01$ python mycrypto.py secret.txt.enc
student@hacklabvm:~/crypto/q01$ ls
mycrypto.py  rsa.encrypted  secret.txt.enc  secret.txt.enc.enc
student@hacklabvm:~/crypto/q01$ cat secret.txt.enc.enc
hacklab_{wrestled-bigwiggedness-banqueteer}
```

Explanation:

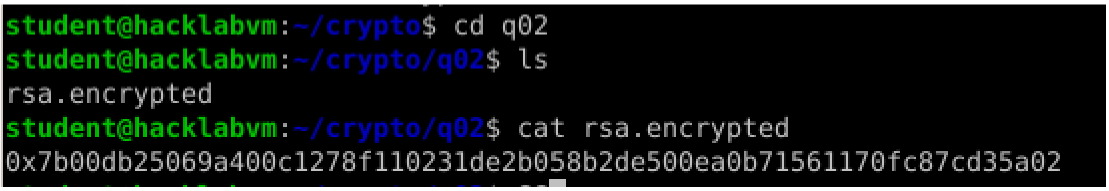
Assuming the default seed value was utilised, the random range should be the same. So, if we use the encrypted file as the input for mycrypto.py, we should get the original input. To which the command is python mycrypto.py secret.txt.enc.

This is a bad encryption because using brute force we could crack it. The key space is sufficiently vast because python has no limitation on the range of integers meaning you can use any number as a key without restriction.

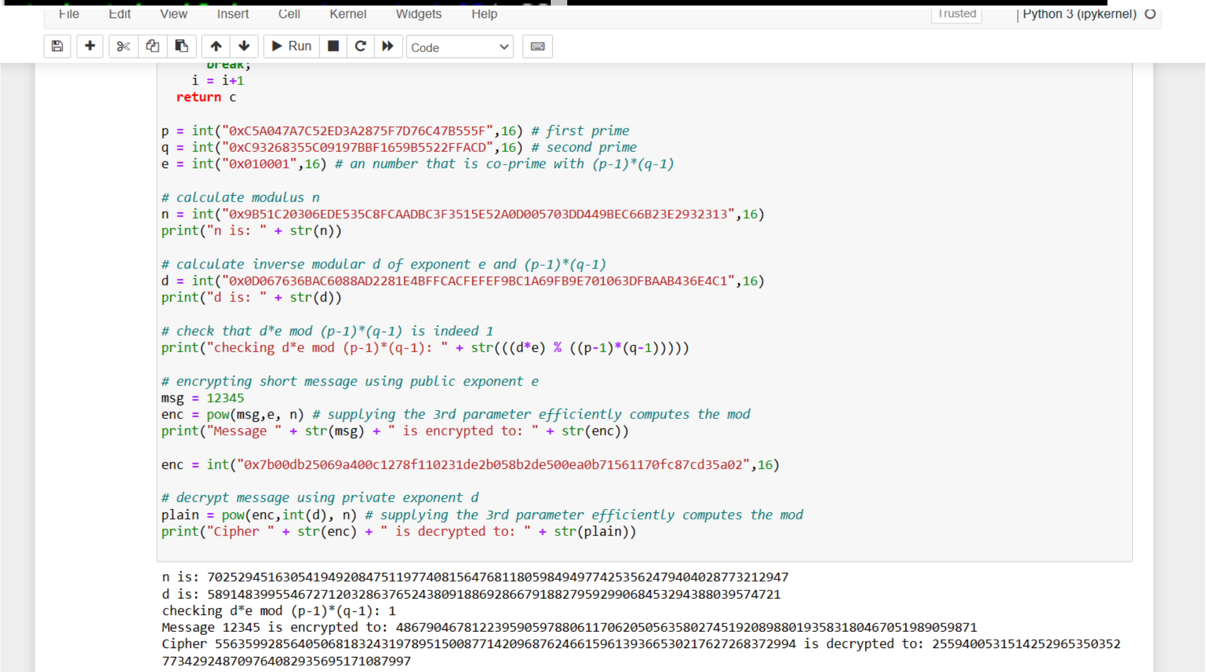
Question 2:

The secret is hacklab_{demonetization}.

Screenshot:

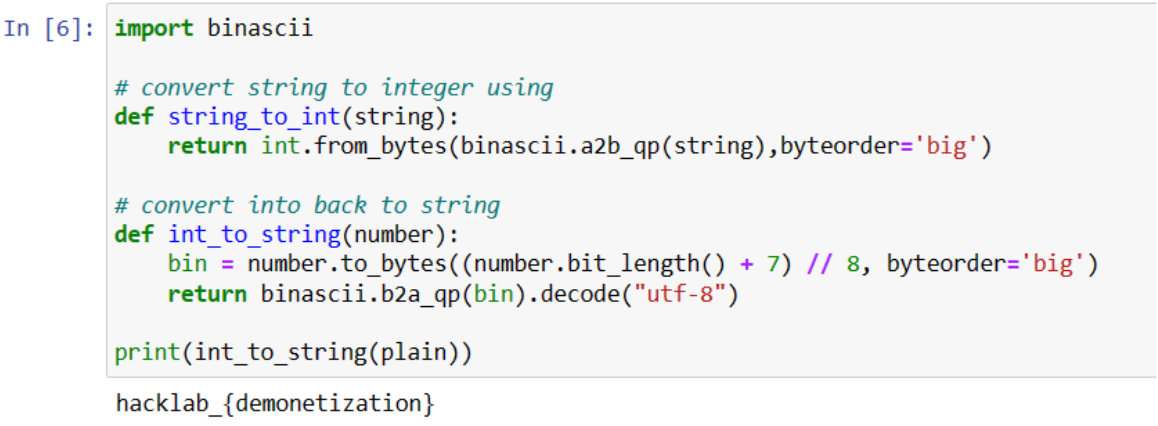


```
student@hacklabvm:~/crypto$ cd q02
student@hacklabvm:~/crypto/q02$ ls
rsa.encrypted
student@hacklabvm:~/crypto/q02$ cat rsa.encrypted
0x7b00db25069a400c1278f110231de2b058b2de500ea0b71561170fc87cd35a02
```



```
        break;
    i = i+1
  return c

p = int("0xC5A047A7C52ED3A2875F7D76C47B555F",16) # first prime
q = int("0xC93268355C09197BBF1659B5522FFACD",16) # second prime
e = int("0x010001",16) # an number that is co-prime with (p-1)*(q-1)

# calculate modulus n
n = int("0x9B51C20306EDE535C8FCAADBC3F3515E52A0D005703DD449BEC66B23E2932313",16)
print("n is: " + str(n))

# calculate inverse modular d of exponent e and (p-1)*(q-1)
d = int("0x0D067636BAC6088AD2281E4BFFCACFEFEF9BC1A69FB9E701063DFBAAB436E4C1",16)
print("d is: " + str(d))

# check that d*e mod (p-1)*(q-1) is indeed 1
print("checking d*e mod (p-1)*(q-1): " + str(((d*e) % ((p-1)*(q-1)))))

# encrypting short message using public exponent e
msg = 12345
enc = pow(msg,e, n) # supplying the 3rd parameter efficiently computes the mod
print("Message " + str(msg) + " is encrypted to: " + str(enc))

enc = int("0x7b00db25069a400c1278f110231de2b058b2de500ea0b71561170fc87cd35a02",16)

# decrypt message using private exponent d
plain = pow(enc,int(d), n) # supplying the 3rd parameter efficiently computes the mod
print("Cipher " + str(enc) + " is decrypted to: " + str(plain))
```

```
n is: 70252945163054194920847511977408156476811805984949774253562479404028773212947
d is: 5891483995546727120328637652438091886928667918827959299068453294388039574721
checking d*e mod (p-1)*(q-1): 1
Message 12345 is encrypted to: 48679046781223959059788061170620505635802745192089880193583180467051989059871
Cipher 55635992856405068183243197895150087714209687624661596139366530217627268372994 is decrypted to: 2559400531514252965350352
773429248709764082935695171087997
```



```
In [6]: import binascii

        # convert string to integer using
        def string_to_int(string):
            return int.from_bytes(binascii.a2b_qp(string),byteorder='big')

        # convert into back to string
        def int_to_string(number):
            bin = number.to_bytes((number.bit_length() + 7) // 8, byteorder='big')
            return binascii.b2a_qp(bin).decode("utf-8")

        print(int_to_string(plain))

        hacklab_{demonetization}
```

Explanation:

I used modified the code provided from the question with the provided values of p,q,e,n,d and used got the enc value from the cat command.

Question 3:

Screenshot:

Explanation:

These are the commands that should be run:

# dd if=original.bmp count=54 ibs=1 >> out.bmp

# dd if=encrypted.bmp skip=54 ibs=1 >> out.bmp

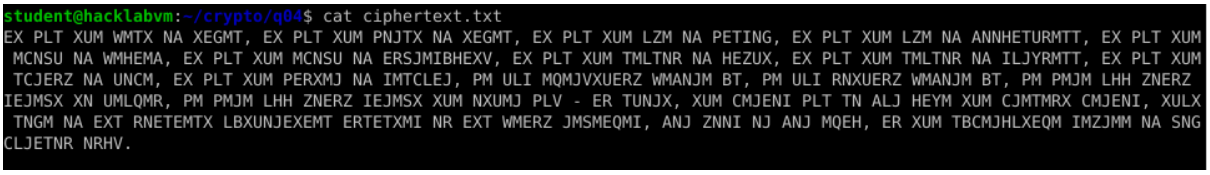
out.bmp

A similar question was presented in the workshop where we were advised to take this approach.

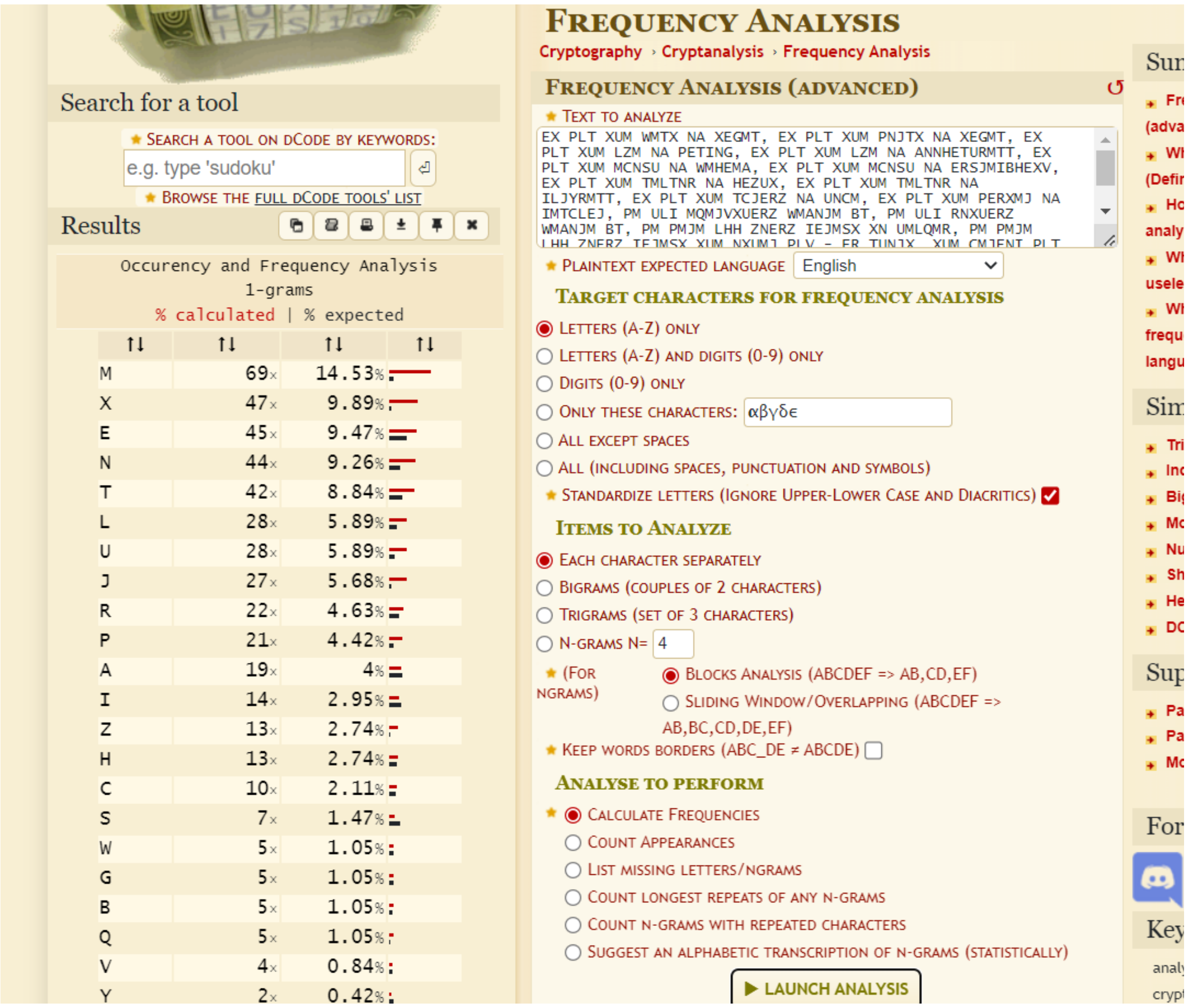
Question 4:

The plain text was: IT WAS THE BEST OF TIMES, IT WAS THE WORST OF TIMES, IT WAS THE AGE OFWISDOM, IT WAS THE AGE OF FOOLISHNESS, IT WAS THE EPOCH OF BELIEF, IT WASTHE EPOCH OF INCREDULITY, IT WAS THE SEASON OF LIGHT, IT WAS THE SEASON OFDARKNESS, IT WAS THE SPRING OF HOPE, IT WAS THE WINTER OF DESPAIR, WE HADEVERYTHING BEFORE US, WE HAD NOTHING BEFORE US, WE WERE ALL GOINGDIRECT TO HEAVEN, WE WERE ALL GOING DIRECT THE OTHER WAY - IN SHORT, THEPERIOD WAS SO FAR LIKE THE PRESENT PERIOD, THAT SOME OF ITS NOISIESTAUTHORITIES INSISTED ON ITS BEING RECEIVED, FOR GOOD OR FOR EVIL, IN THESUPERLATIVE DEGREE OF COMPARISON ONLY.

Screenshot:



```
student@hacklabvm:~/crypto/q04$ cat ciphertext.txt
EX PLT XUM WMTX NA XEGMT, EX PLT XUM PNJTX NA XEGMT, EX PLT XUM LZM NA PETING, EX PLT XUM LZM NA ANNHETURMTT, EX PLT XUM
 MCNSU NA WMHEMA, EX PLT XUM MCNSU NA ERSJMIBHEXV, EX PLT XUM TMLTNR NA HEZUX, EX PLT XUM TMLTNR NA ILJYRMTT, EX PLT XUM
 TCJERZ NA UNCM, EX PLT XUM PERXMJ NA IMTCLEJ, PM ULI MQMJVXUERZ WMANJM BT, PM ULI RNXUERZ WMANJM BT, PM PMJM LHH ZNERZ
IEJMSX XN UMLQMR, PM PMJM LHH ZNERZ IEJMSX XUM NXUMJ PLV - ER TUNJX, XUM CMJENI PLT TN ALJ HEYM XUM CJMTMRX CMJENI, XULX
 TNGM NA EXT RNETEMTX LBXUNJEXEMT ERTETXMI NR EXT WMERZ JMSMEQMI, ANJ ZNNI NJ ANJ MQEH, ER XUM TBCMJHLXEQM IMZJMM NA SNG
CLJETNR NRHV.
```



Search for a tool

★ Search a tool on dCode by keywords:

e.g. type 'sudoku'

★ Browse the full dCode tools' list

Results

Occurency and Frequency Analysis
1-grams
% calculated | % expected

↑↓	↑↓	↑↓	↑↓
M	69×	14.53%	
X	47×	9.89%	
E	45×	9.47%	
N	44×	9.26%	
T	42×	8.84%	
L	28×	5.89%	
U	28×	5.89%	
J	27×	5.68%	
R	22×	4.63%	
P	21×	4.42%	
A	19×	4%	
I	14×	2.95%	
Z	13×	2.74%	
H	13×	2.74%	
C	10×	2.11%	
S	7×	1.47%	
W	5×	1.05%	
G	5×	1.05%	
B	5×	1.05%	
Q	5×	1.05%	
V	4×	0.84%	
Y	2×	0.42%	

Frequency Analysis

Cryptography › Cryptanalysis › Frequency Analysis

Frequency Analysis (advanced)

★ Text to analyze

```
EX PLT XUM WMTX NA XEGMT, EX PLT XUM PNJTX NA XEGMT, EX
PLT XUM LZM NA PETING, EX PLT XUM LZM NA ANNHETURMTT, EX
PLT XUM MCNSU NA WMHEMA, EX PLT XUM MCNSU NA ERSJMIBHEXV,
EX PLT XUM TMLTNR NA HEZUX, EX PLT XUM TMLTNR NA
ILJYRMTT, EX PLT XUM TCJERZ NA UNCM, EX PLT XUM PERXMJ NA
IMTCLEJ, PM ULI MQMJVXUERZ WMANJM BT, PM ULI RNXUERZ
WMANJM BT, PM PMJM LHH ZNERZ IEJMSX XN UMLQMR, PM PMJM
LHH ZNERZ IEJMSX XUM NXUMJ PLV - ER TUNJX, XUM CMJENI PLT
```

★ Plaintext expected language: English

Target characters for frequency analysis

● Letters (A-Z) only
○ Letters (A-Z) and digits (0-9) only
○ Digits (0-9) only
○ Only these characters: αβγδε
○ All except spaces
○ All (including spaces, punctuation and symbols)
★ Standardize letters (Ignore Upper-Lower Case and Diacritics) ☑

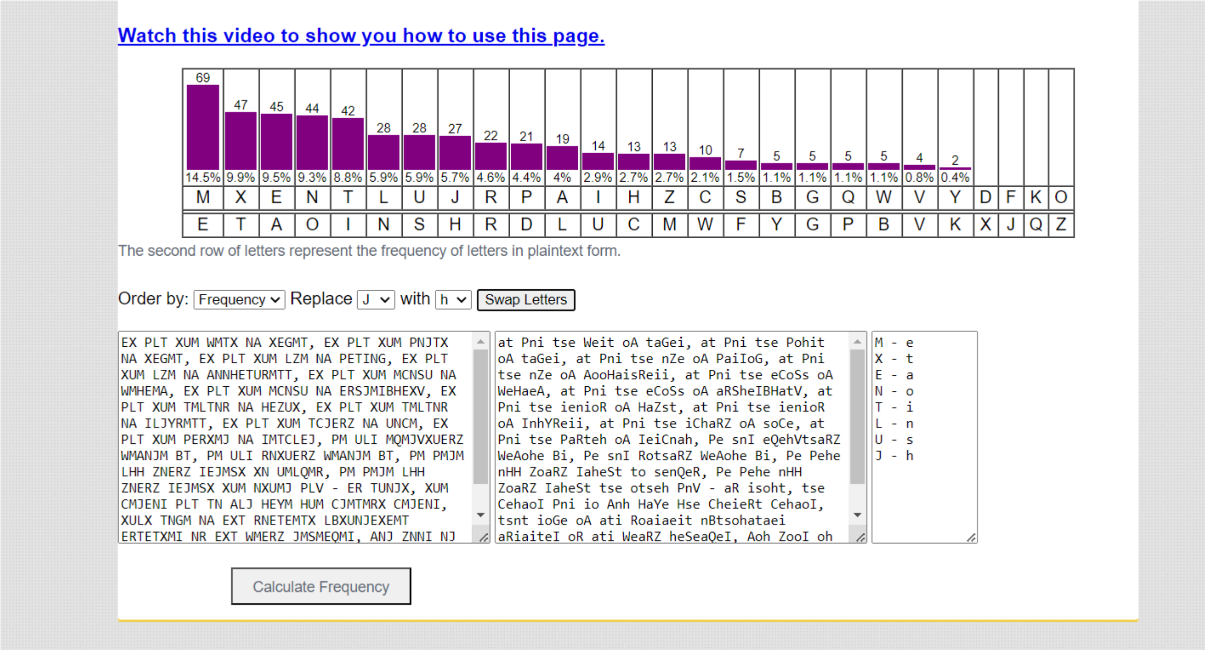
Items to analyze

● Each character separately
○ Bigrams (couples of 2 characters)
○ Trigrams (set of 3 characters)
○ N-grams N= 4
★ (For ngrams) ● Blocks analysis (ABCDEF => AB,CD,EF)
○ Sliding Window/Overlapping (ABCDEF => AB,BC,CD,DE,EF)
★ Keep words borders (ABC_DE ≠ ABCDE) □

Analyse to perform

★ ● Calculate frequencies
○ Count appearances
○ List missing letters/ngrams
○ Count longest repeats of any n-grams
○ Count n-grams with repeated characters
○ Suggest an alphabetic transcription of n-grams (statistically)

▶ Launch analysis



Watch this video to show you how to use this page.

69	47	45	44	42	28	28	27	22	21	19	14	13	13	10	7	5	5	5	5	4	2				
14.5%	9.9%	9.5%	9.3%	8.8%	5.9%	5.9%	5.7%	4.6%	4.4%	4%	2.9%	2.7%	2.7%	2.1%	1.5%	1.1%	1.1%	1.1%	1.1%	0.8%	0.4%				
M	X	E	N	T	L	U	J	R	P	A	I	H	Z	C	S	B	G	Q	W	V	Y	D	F	K	O
E	T	A	O	I	N	S	H	R	D	L	U	C	M	W	F	Y	G	P	B	V	K	X	J	Q	Z

The second row of letters represent the frequency of letters in plaintext form.

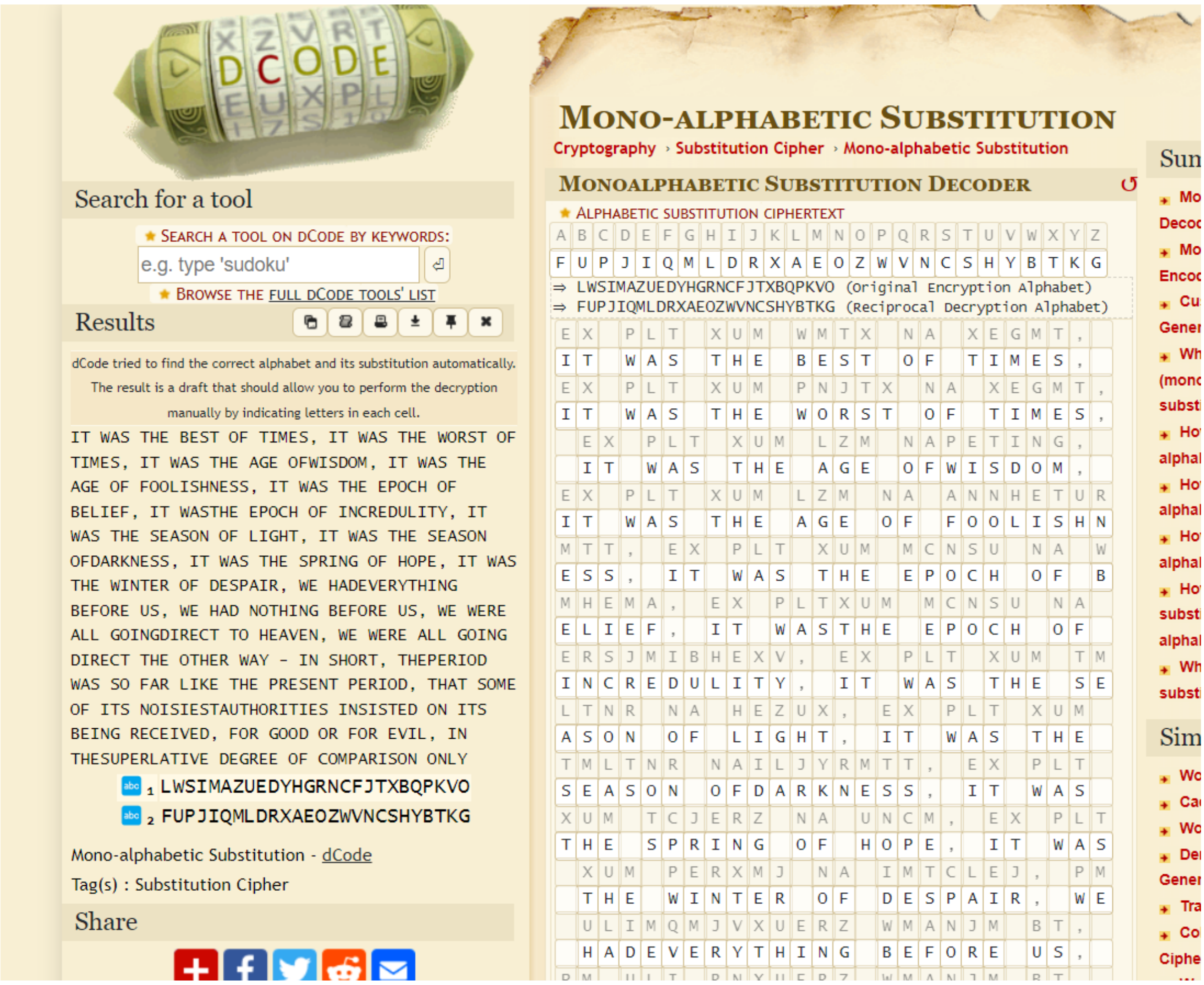
Order by: Frequency Replace J with h Swap Letters

```
EX PLT XUM WMTX NA XEGMT, EX PLT XUM PNJTX
NA XEGMT, EX PLT XUM LZM NA PETING, EX PLT
XUM LZM NA ANNHETURMTT, EX PLT XUM MCNSU NA
WMHEMA, EX PLT XUM MCNSU NA ERSJMIBHEXV, EX
PLT XUM TMLTNR NA HEZUX, EX PLT XUM TMLTNR
NA ILJYRMTT, EX PLT XUM TCJERZ NA UNCM, EX
PLT XUM PERXMJ NA IMTCLEJ, PM ULI MQMJVXUERZ
WMANJM BT, PM ULI RNXUERZ WMANJM BT, PM PMJM
LHH ZNERZ IEJMSX XN UMLQMR, PM PMJM LHH
ZNERZ IEJMSX XUM NXUMJ PLV - ER TUNJX, XUM
CMJENI PLT TN ALJ HEYM HUM CJMTMRX CMJENI,
XULX TNGM NA EXT RNETEMTX LBXUNJEXEMT
ERTETXMI NR EXT WMERZ JMSMEQMI, ANJ ZNNI NJ
```

```
at Pni tse Weit oA taGei, at Pni tse Pohit
oA taGei, at Pni tse nZe oA PaiIoG, at Pni
tse nZe oA AooHaisReii, at Pni tse eCoSs oA
WeHaeA, at Pni tse eCoSs oA aRSheIBHatV, at
Pni tse ienioR oA HaZst, at Pni tse ienioR
oA InhYReii, at Pni tse iChaRZ oA soCe, at
Pni tse PaRteh oA IeiCnah, Pe snI eQehVtsaRZ
WeAohe Bi, Pe snI RotsaRZ WeAohe Bi, Pe Pehe
nHH ZoaRZ IaheSt to senQeR, Pe Pehe nHH
ZoaRZ IaheSt tse otseh PnV - aR isoht, tse
CehaoI Pni io Anh HaYe Hse CheieRt CehaoI,
tsnt ioGe oA ati Roaiaeit nBtsohataei
aRiaiteI oR ati WeaRZ heSeaQeI, Aoh ZooI oh
```

```
M - e
X - t
E - a
N - o
T - i
L - n
U - s
J - h
```

Calculate Frequency



Search for a tool

★ SEARCH A TOOL ON DCODE BY KEYWORDS:

e.g. type 'sudoku'

★ BROWSE THE FULL DCODE TOOLS' LIST

Results

dCode tried to find the correct alphabet and its substitution automatically. The result is a draft that should allow you to perform the decryption manually by indicating letters in each cell.

IT WAS THE BEST OF TIMES, IT WAS THE WORST OF TIMES, IT WAS THE AGE OFWISDOM, IT WAS THE AGE OF FOOLISHNESS, IT WAS THE EPOCH OF BELIEF, IT WASTHE EPOCH OF INCREDULITY, IT WAS THE SEASON OF LIGHT, IT WAS THE SEASON OFDARKNESS, IT WAS THE SPRING OF HOPE, IT WAS THE WINTER OF DESPAIR, WE HADEVERYTHING BEFORE US, WE HAD NOTHING BEFORE US, WE WERE ALL GOINGDIRECT TO HEAVEN, WE WERE ALL GOING DIRECT THE OTHER WAY - IN SHORT, THEPERIOD WAS SO FAR LIKE THE PRESENT PERIOD, THAT SOME OF ITS NOISIESTAUTHORITIES INSISTED ON ITS BEING RECEIVED, FOR GOOD OR FOR EVIL, IN THESUPERLATIVE DEGREE OF COMPARISON ONLY

1 LWSIMAZUEDYHGRNCFJTXBQPKVO

2 FUPJIQMLDRXAEOZWVNCSHYBTKG

Mono-alphabetic Substitution - dCode

Tag(s) : Substitution Cipher

Share

Mono-alphabetic Substitution

Cryptography › Substitution Cipher › Mono-alphabetic Substitution

Monoalphabetic Substitution Decoder

★ ALPHABETIC SUBSTITUTION CIPHERTEXT

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
F	U	P	J	I	Q	M	L	D	R	X	A	E	O	Z	W	V	N	C	S	H	Y	B	T	K	G

⇒ LWSIMAZUEDYHGRNCFJTXBQPKVO (Original Encryption Alphabet)
⇒ FUPJIQMLDRXAEOZWVNCSHYBTKG (Reciprocal Decryption Alphabet)

```
E X   P L T   X U M   W M T X   N A   X E G M T ,
I T   W A S   T H E   B E S T   O F   T I M E S ,
E X   P L T   X U M   P N J T X   N A   X E G M T ,
I T   W A S   T H E   W O R S T   O F   T I M E S ,
  E X   P L T   X U M   L Z M   N A P E T I N G ,
  I T   W A S   T H E   A G E   O F W I S D O M ,
E X   P L T   X U M   L Z M   N A   A N N H E T U R
I T   W A S   T H E   A G E   O F   F O O L I S H N
M T T ,   E X   P L T   X U M   M C N S U   N A   W
E S S ,   I T   W A S   T H E   E P O C H   O F   B
M H E M A ,   E X   P L T X U M   M C N S U   N A
E L I E F ,   I T   W A S T H E   E P O C H   O F
E R S J M I B H E X V ,   E X   P L T   X U M   T M
I N C R E D U L I T Y ,   I T   W A S   T H E   S E
L T N R   N A   H E Z U X ,   E X   P L T   X U M
A S O N   O F   L I G H T ,   I T   W A S   T H E
T M L T N R   N A I L J Y R M T T ,   E X   P L T
S E A S O N   O F D A R K N E S S ,   I T   W A S
X U M   T C J E R Z   N A   U N C M ,   E X   P L T
T H E   S P R I N G   O F   H O P E ,   I T   W A S
  X U M   P E R X M J   N A   I M T C L E J ,   P M
  T H E   W I N T E R   O F   D E S P A I R ,   W E
  U L I M Q M J V X U E R Z   W M A N J M   B T ,
  H A D E V E R Y T H I N G   B E F O R E   U S ,
```

Explanation:

After finding cipher text in hacklab, I did a frequency analysis tool. Then after trying manually for some time. I used the Mono-alphabetic Substitution tool to convert it to plain text and finding key.

Question 5:

Screenshot:

Explanation: