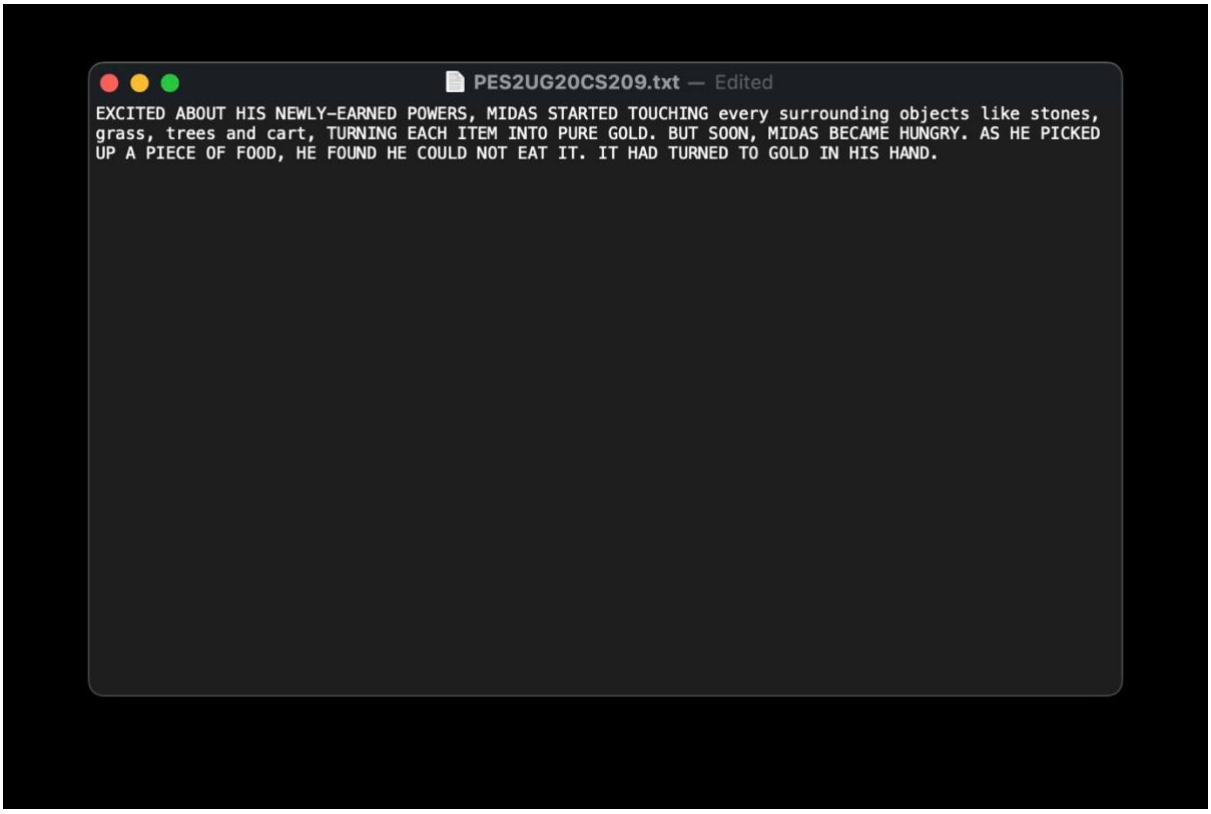
Applied Cryptography
UE20CS214

Name	Naman Choudhary
SRN	PES2UG20CS209
Section	D

1. **Create and display a file SRN.txt with following contents:**

   EXCITED ABOUT HIS NEWLY-EARNED POWERS, MIDAS STARTED TOUCHING every surrounding objects like stones, grass, trees and cart, TURNING EACH ITEM INTO PURE GOLD. BUT SOON, MIDAS BECAME HUNGRY. AS HE PICKED UP A PIECE OF FOOD, HE FOUND HE COULD NOT EAT IT. IT HAD TURNED TO GOLD IN HIS HAND.



2. **In file SRN.txt, convert uppercase letters to lowercase and find the frequencies of following words:**

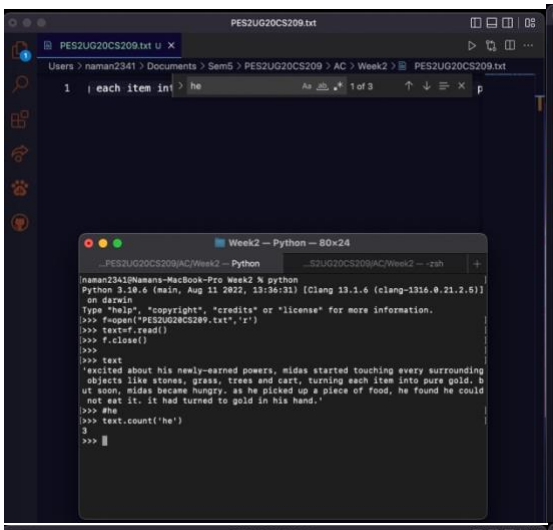
   a. **He**
   b. **H**

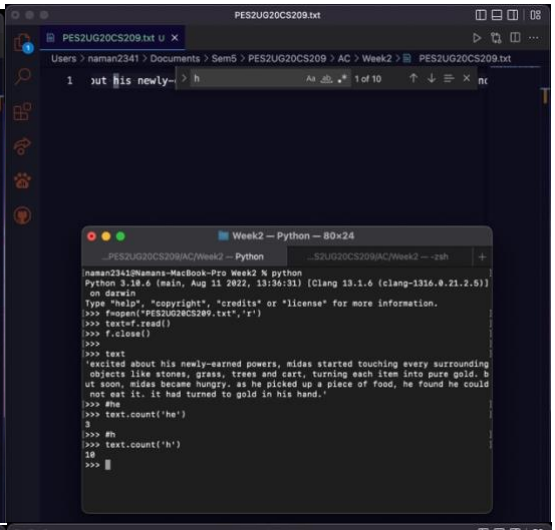
c. **Ed**
d. **Oo**
e. **A**
f. **As**

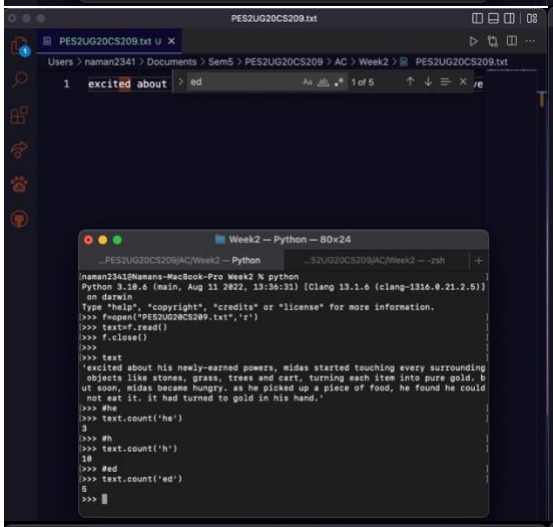
```
Week2 — -zsh — 80×24
...5/PES2UG20CS209/AC/Week2 — -zsh    ...S2UG20CS209/AC/Week2 — -zsh
Last login: Sun Sep  4 22:40:53 on ttys000
naman2341@Namans-MacBook-Pro ~ % cd Documents/Sem5/PES2UG20CS209/AC/Week2/
naman2341@Namans-MacBook-Pro Week2 % python3
Python 3.10.6 (main, Aug 11 2022, 13:36:31) [Clang 13.1.6 (clang-1316.0.21.2.5)]
 on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> text=open("PES2UG20CS209.txt",'r').read()
>>> text
'EXCITED ABOUT HIS NEWLY-EARNED POWERS, MIDAS STARTED TOUCHING every surrounding
 objects like stones, grass, trees and cart, TURNING EACH ITEM INTO PURE GOLD. B
UT SOON, MIDAS BECAME HUNGRY. AS HE PICKED UP A PIECE OF FOOD, HE FOUND HE COULD
 NOT EAT IT. IT HAD TURNED TO GOLD IN HIS HAND.'
>>> open("PES2UG20CS209.txt",'w').write(text.lower())
286
>>> exit
Use exit() or Ctrl-D (i.e. EOF) to exit
>>> exit()
naman2341@Namans-MacBook-Pro Week2 % cat PES2UG20CS209.txt
excited about his newly-earned powers, midas started touching every surrounding
objects like stones, grass, trees and cart, turning each item into pure gold. bu
t soon, midas became hungry. as he picked up a piece of food, he found he could
not eat it. it had turned to gold in his hand.%
naman2341@Namans-MacBook-Pro Week2 %
```

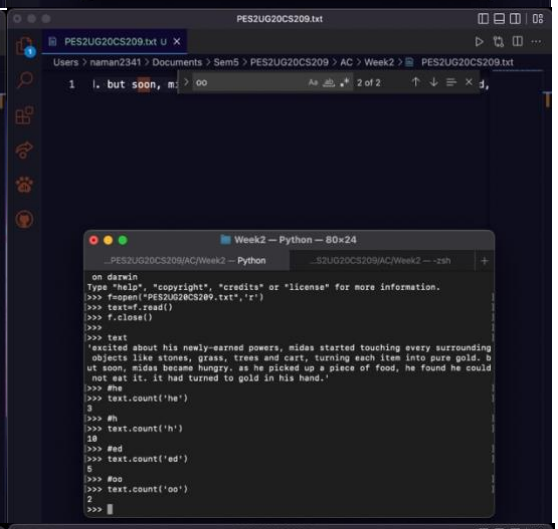
```
# Frequency Analysis
Letter Combination          Count
===================         =====
he                          3
h                           10
ed                          5
oo                          2
a                           15
as                          4
```

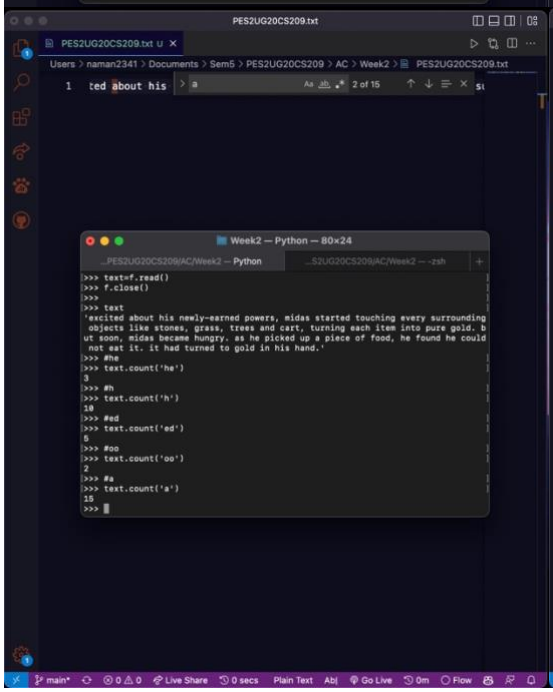
3. Highlighting the words

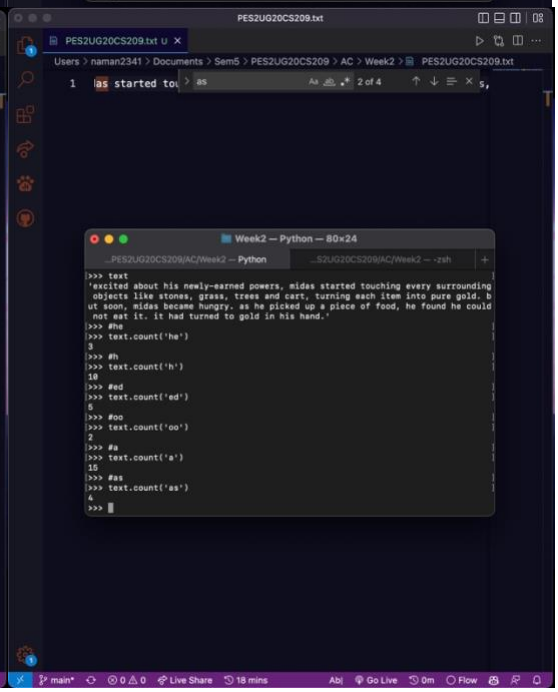












4. Generate the substitution cipher key.

```python
def generate_key(alphabet_string):
    import random as r
    l = list(alphabet_string)
    r.shuffle(l)
    return ''.join(l)
```

Key Generation:

```
Week2 — Python — 80×24
naman2341@Namans-MacBook-Pro SSs % python
Python 3.10.6 (main, Aug 11 2022, 13:36:31) [Clang 13.1.6 (clang-1316.0.21.2.5)]
 on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
naman2341@Namans-MacBook-Pro SSs % cd ../
naman2341@Namans-MacBook-Pro Week2 % python
Python 3.10.6 (main, Aug 11 2022, 13:36:31) [Clang 13.1.6 (clang-1316.0.21.2.5)]
 on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from util import *
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'vicwjeuhprqsdtgmxlbknayofz'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'qegsxaplotycwihnrjfmkdvubz'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'tzryxqgbosvnwaimkudplcjhef'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'hlqcgpswneotdubjmxriyazvkf'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'bgpvwhrtynozujiscaqdlxkfem'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'qlmtjucrbvehxkawfiogzdnsyp'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
```

Key: sadnmevjufzckptihqbrlxogwy

5. Generate the cipher text using the key generated in question 3.

encrypt.py > ...

```python
#! /usr/bin/env python

f=open('new_text.txt','r')
plaintext=f.read()
f.close()

ciphertext=''

alphabet='abcdefghijklmnopqrstuvwxyz'
key='sadnmevjufzckptihqbrlxogwy'

for i in range(len(plaintext)):
    if plaintext[i] not in alphabet:
        ciphertext+=plaintext[i]
    else:
        index_in_alphabet=alphabet.index(plaintext[i])
        ciphertext+=key[index_in_alphabet]

print(ciphertext)
```

Ciphertext:

Week2 — -zsh — 85×24

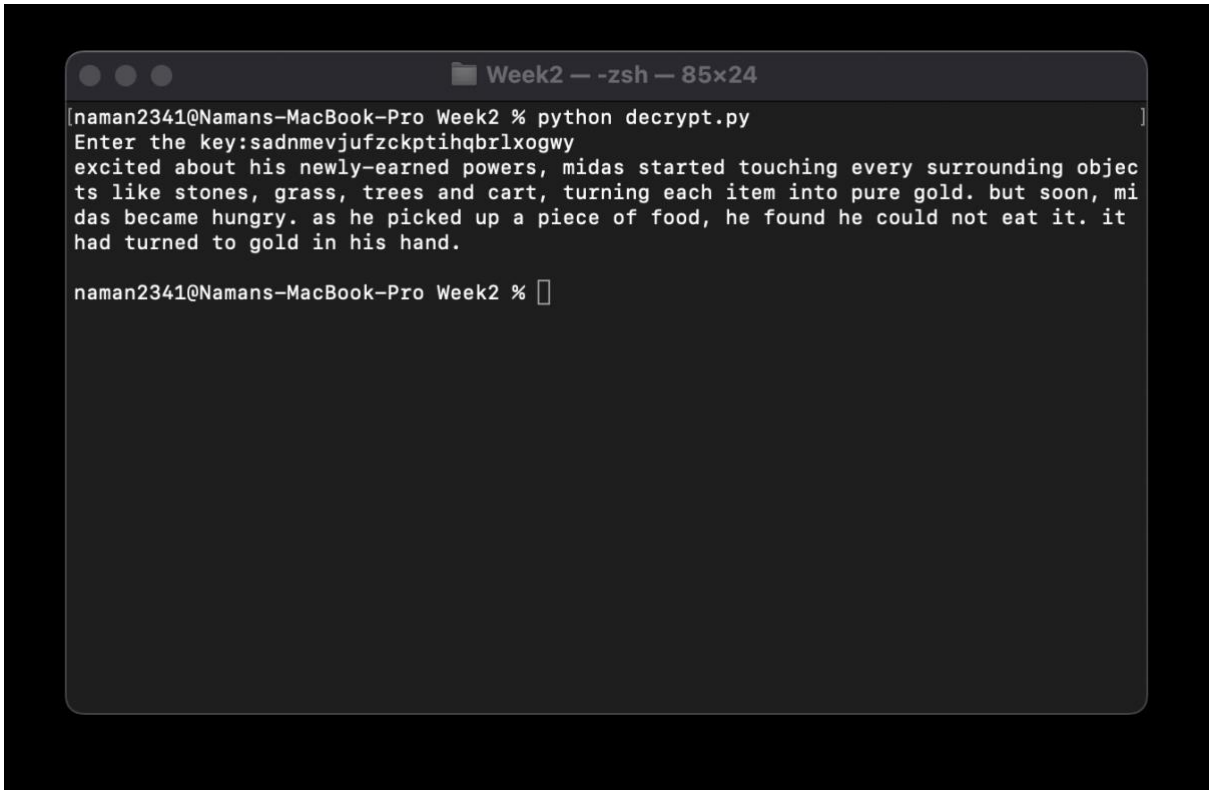
```
naman2341@Namans-MacBook-Pro Week2 % python3 encrypt.py > PES2UG20CS209_encrypted.txt
naman2341@Namans-MacBook-Pro Week2 % cat PES2UG20CS209_encrypted.txt
mgdurmn satlr jub pmocw-msqpmn itomqb, kunsb brsqrmn rtldjupv mxmqw blqqtlpnupv tafmd
rb cuzm brtpmb, vqsbb, rqmmb spn dsqr, rlqpupv msdj urmk uprt ilqm vtcn. alr bttp, ku
nsb amdskm jlpvqw. sb jm iudzmn li s iumdm te ettn, jm etlpn jm dtlcn ptr msr ur. ur
jsn rlqpmn rt vtcn up jub jspn.
naman2341@Namans-MacBook-Pro Week2 %
```

6. Decrypt the cipher text back to plain text.

decrypt.py > ...

```python
#!/usr/bin/env python3

from util import check_key_validity
f=open('new_encrypt.txt','r')
ciphertext=f.read()
f.close()

plaintext=''

alphabet = 'abcdefghijklmnopqrstuvwxyz'

key= input("Enter the key:")

if not check_key_validity(key,alphabet):
    print('Invalid key')
    exit(1)
else:
    for i in range(len(ciphertext)):
        if ciphertext[i] not in key:
            plaintext+=ciphertext[i]
        else:
            index_in_key=key.index(ciphertext[i])
            plaintext+=alphabet[index_in_key]
    print(plaintext)
```

Decrypted ciphertext to plaintext:



7. Suppose the input file is :

> ***Hungry, Midas groaned, "I'll starve! Perhaps this was not such an excellent wish after all!"...***

Ciphertext:

```
Week2 — -zsh — 85×24
naman2341@Namans-MacBook-Pro Week2 % python encrypt.py
Hlpvqw, Munsb vqtspmn, "I'cc brsqxm! Pmqjsib rjub osb ptr bldj sp mgdmccmpr oubj serm
q scc!"...
naman2341@Namans-MacBook-Pro Week2 % python encrypt.py>new_encrypt.txt
naman2341@Namans-MacBook-Pro Week2 %
```

Decrypted ciphertext to plaintext:

```
Week2 — -zsh — 85×24
naman2341@Namans-MacBook-Pro Week2 % python decrypt.py
Enter the key:sadnmevjufzckptihqbrlxogwy
Hungry, Midas groaned, "I'll starve! Perhaps this was not such an excellent wish afte
r all!"...

naman2341@Namans-MacBook-Pro Week2 %
```

Comments:

We note that the newly generated cipher text is shorter than the previously generated ciphertext, and it appears to be the same length as the input plaintext, but this type of encryption is not safe, as modern computers can easily crack this with frequency distribution of letters, or simply by bruteforce attack(long but not impossible).