Assignment 2

General Instructions:

1. Deadline: Tuesday, October 15at 11:59 pm.
2. This assignment is worth 10 pts of your course grade.
3. Download the file: “template_A2.py” and rename it to “solution_A2.py”. This should be strictly observed for the testing file to work.
4. At the top of the file: edit the following header (Any submission without this header will be rejected):

```
#-------------------------
# Your Name and ID
# CP460 (Fall 2019)
# Assignment 2
#-------------------------
```

5. There are two other files: “utilities_A2.py” and “test_A2.py”. Download the two files into the same folder as “solution_A2.py”.
6. Download the ciphertext, plaintext and dictionary files into the same folder
7. You may only edit the solution_A2.py. The “utilities_A2.py” and “test_A2.py” files should not be changed. If you need to add any other utility function, add it into the solution_A2.py file.
8. Make sure to go through the utilities file. You do not need to create a function that is already provided to you.
9. Every new function you create should have a header outlining input parameters, return values and a description. similar to the following:

```
#-----------------------------------------------------
# Parameters:   ciphertext(string)
#               key (none)
# Return:       plaintext (string)
# Description:  Decryption using Polybius Square
#----------------------------------------------------
```

10. At the end, submit ONLY the solution_A2.py file.

Q1: Vigenere Cipher Implementation (1.6 pts)

In class, the code for the Vigenere cipher (version 1) using a key word of a single character was provided. In this task, you will modify the encryption and decryption schemes such that the autokey is a given phrase, i.e. of length two or more characters. We will call this Vigenere Cipher (version 2).

Write the encryption function `e_vigenere2(plaintext, key)` that would encrypt any given plaintext using the Vigenere Cipher through an autokey which has two or more characters. Also, write the decryption function `d_vigenere2(ciphertext, key)` that would perform the reverse process to restore the original plaintext.

A valid ***key*** should be a non-empty string that contains only alpha characters.

It would be easier to take the implementations of `e_vigenere1(plaintext, key)` and `d_vigenere1(plaintext, key)` and edit them according to the new requirements.

Remember that both `e_vigenere1` and `e_vigenere2` are called by `e_vigenere` which calls the proper function depending on the key length. Similarly, `d_vigenere1` and `d_vigenere2` are called by `d_vigenere`.

The `e_vigenere` and `d_vigenere` functions are provided to you. You would need to copy from your class notes `e_vigenere1` and `d_vigenere1` functions, and write your own `e_vigenere2` and `d_vigenere2`.

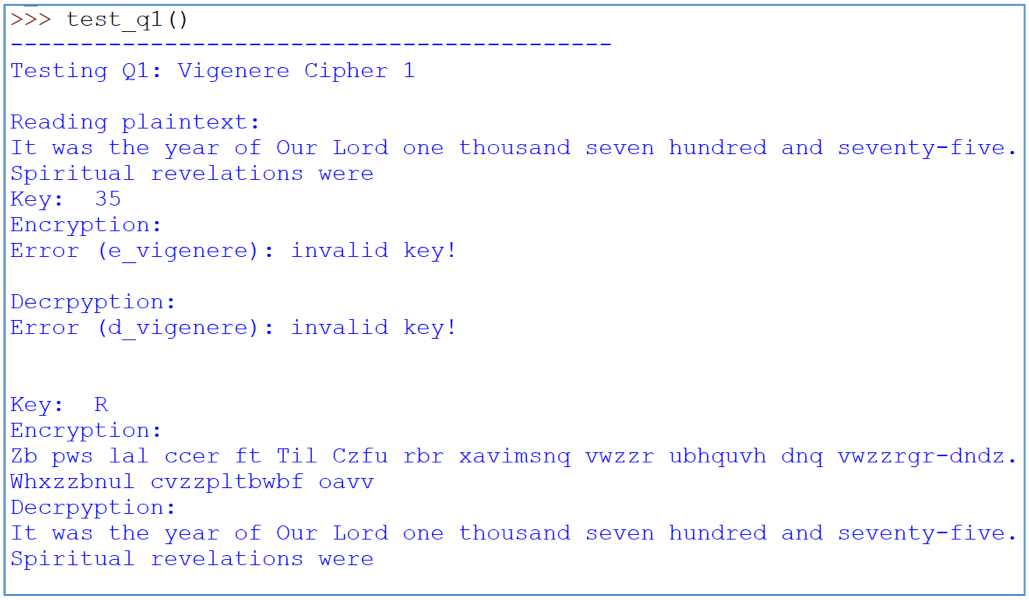
The function descriptions are as follows:

```
#-----------------------------------------------------
# Parameters:   plaintext (string)
#               key (string)
# Return:       ciphertext (string)
# Description:  Encryption using Vigenere cipher
#               Autokey is of length 2 or more alpha characters
#                Non-alpha characters → no substitution
#                Preservers the case of the plaintext characters
#----------------------------------------------------
```

```
def e_vigenere2(plaintext, key):
    # your code here
    return ciphertext

#----------------------------------------------------
# Parameters:   ciphertext (string)
#               key (string)
# Return:       plaintext (string)
# Description:  Decryption using Vigenere cipher
#               Autokey is of length 2 or more alpha characters
#                Non-alpha characters → no substitution
#                Preservers the case of the plaintext characters
#---------------------------------------------------
def d_vigenere2(ciphertext, key):
    # your code here
    return plaintext
```

Below are the results of executing the testing module:



```
>>> test_q1()
-------------------------------------------
Testing Q1: Vigenere Cipher 1

Reading plaintext:
It was the year of Our Lord one thousand seven hundred and seventy-five.
Spiritual revelations were
Key:  35
Encryption:
Error (e_vigenere): invalid key!

Decrpyption:
Error (d_vigenere): invalid key!


Key:  R
Encryption:
Zb pws lal ccer ft Til Czfu rbr xavimsnq vwzzr ubhquvh dnq vwzzrgr-dndz.
Whxzzbnul cvzzpltbwbf oavv
Decrpyption:
It was the year of Our Lord one thousand seven hundred and seventy-five.
Spiritual revelations were
```

```
Key:  ON
Encryption:
Wg kng gvr mroe cs Chf Ycer bbr huchgnbq grjrb uiaresq oar fsisahl-tvjr.
Gcwewginz esisyogwbbf krfr
Decrpyption:
It was the year of Our Lord one thousand seven hundred and seventy-five.
Spiritual revelations were

Key:  SIT
Encryption:
Ab psa mzm rwik gn Hmz Egzw gvx lphmatfl lwdxf pnflkwl tfl lwdxfbr-xqow.
Aiazblctd zxnmesbbgvl omkw
Decrpyption:
It was the year of Our Lord one thousand seven hundred and seventy-five.
Spiritual revelations were

Key:  FORD
Encryption:
Nh ndx hyh dsru tt Fxw Zfui ceh yvfxxoeg xsmhs vlqifvg fbu vjjvqym-wlas.
Jsnfzwzoc ujjvofhzrsg nhws
Decrpyption:
It was the year of Our Lord one thousand seven hundred and seventy-five.
Spiritual revelations were
```

```
Key:  PEACE
Encryption:
Xx wcw ile aipv oh Sjv Lqvs sng xwsuuech sgztr hwrsvef ech sgztrta-jxze.
Utxvivypp rgztpavmdrs yigi
Decrpyption:
It was the year of Our Lord one thousand seven hundred and seventy-five.
Spiritual revelations were

Key:  Jellyfish
Encryption:
Rx hlq ypw fnec zd Tcj Sxvo zlj bzvdwlyb xmnlw lfybwmv hwh dptjvlf-omgp.
Quqjpcylw pjdwsjxtzlx ewyn
Decrpyption:
It was the year of Our Lord one thousand seven hundred and seventy-five.
Spiritual revelations were

Key:  ENVIORNMENT
Encryption:
Mg rig kuq crtv ba Wii Yavq hrr opclfmrq liizv vlapvrw eay asmrzxl-ymiz.
Adzeuxhtp ezdscnfmbgw jzzs
Decrpyption:
It was the year of Our Lord one thousand seven hundred and seventy-five.
Spiritual revelations were

-------------------------------------------
```

Q2: Vigenere Cryptanalysis Utilities (2.4 pts)

In order to execute a successful Vigenere cryptanalysis, a cryptanalysis would need several automated tools (functions) at hand. In this task, you will develop six of these tools:

```
#----------------------------------------------------------------------------
# Parameters:   text (string)
#               size (int)
# Return:       blocks: list of strings
# Description:  Break a given string into blocks of strings of given size
#               Result is provided in a list
#----------------------------------------------------------------------------
def text_to_blocks(text,size):
    # your code here
    return blocks
```

```
#-----------------------------------
# Parameters:   text (string)
# Return:       modifiedText (string)
# Description:  Removes all non-alpha characters from the given string
#               Returns a string of only alpha characters (upper case)
#-----------------------------------
def remove_nonalpha(text):
    # your code here
    return modifiedText
```

```
#----------------------------------------------------------------------------
# Parameters:   blocks: list of strings
# Return:       baskets: list of strings
# Description:  Assume all blocks have same size = n (other than last block)
#               Create n baskets
#               In basket[i] put character #i from each block
#----------------------------------------------------------------------------
def blocks_to_baskets(blocks):
    # your code here
    return baskets
```

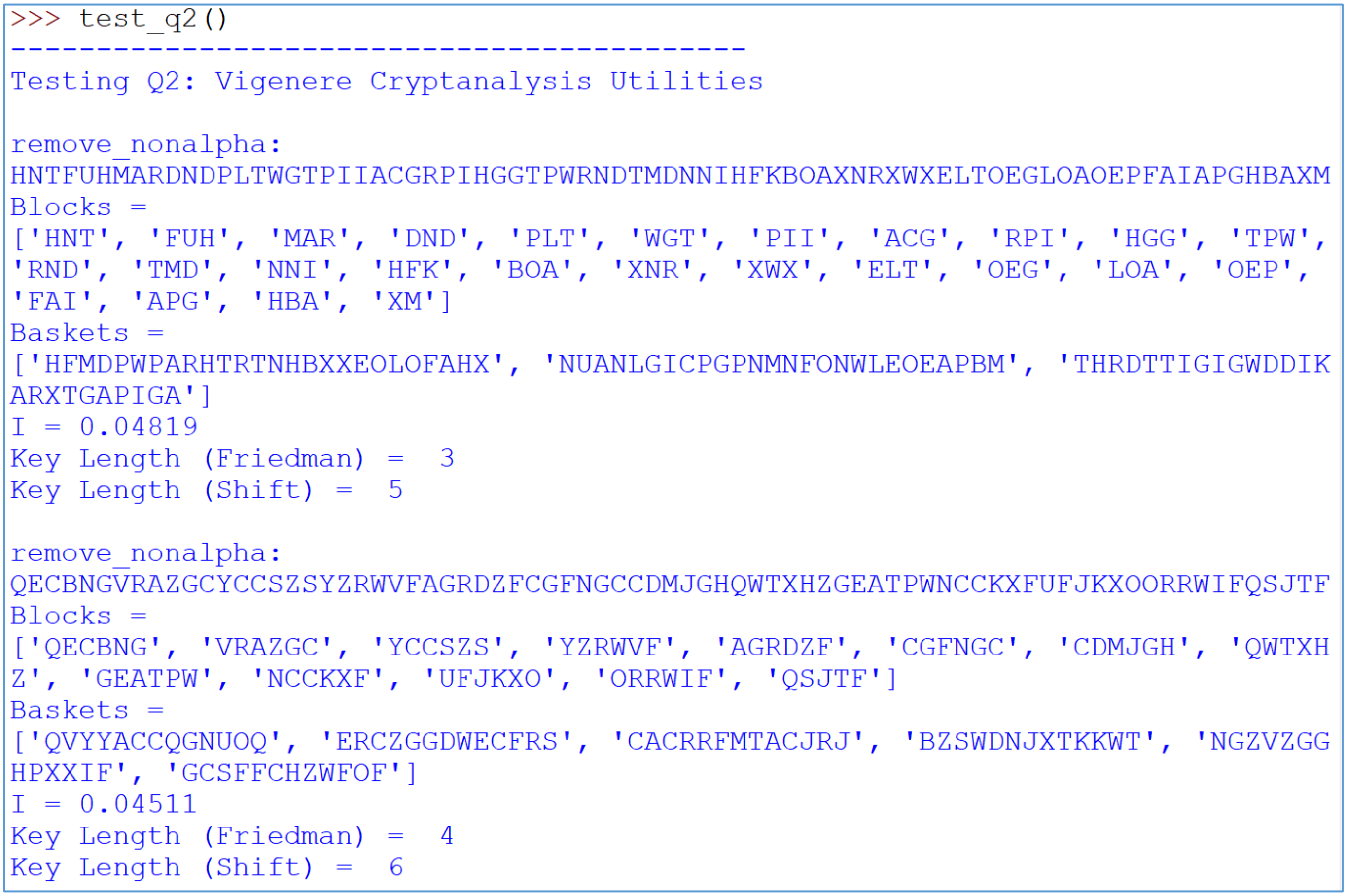
```
#----------------------------------------------------------------------------
# Parameters:   ciphertext(string)
# Return:       I (float): Index of Coincidence
# Description:  Computes and returns the index of coincidence
#               for a given text
#----------------------------------------------------------------------------
def get_indexOfCoin(ciphertext):
    #your code here
    return I
```

```
#----------------------------------------------------------------
# Parameters:   ciphertext(string)
# Return:       k (int) key length
# Description:  Uses Friedman's test to compute key length
#               returns key length rounded to nearest integer
#---------------------------------------------------------------
def getKeyL_friedman(ciphertext):
    # your code here
    return k
```

```
#----------------------------------------------------------------
# Parameters:   ciphertext(string)
# Return:       key (int)
# Description:  Uses the Ciphertext Shift method to compute key length
#               Attempts key lengths 1 to 20
#---------------------------------------------------------------
def getKeyL_shift(ciphertext):
   # your code here
   return k
```

Below are the results of executing the testing module:

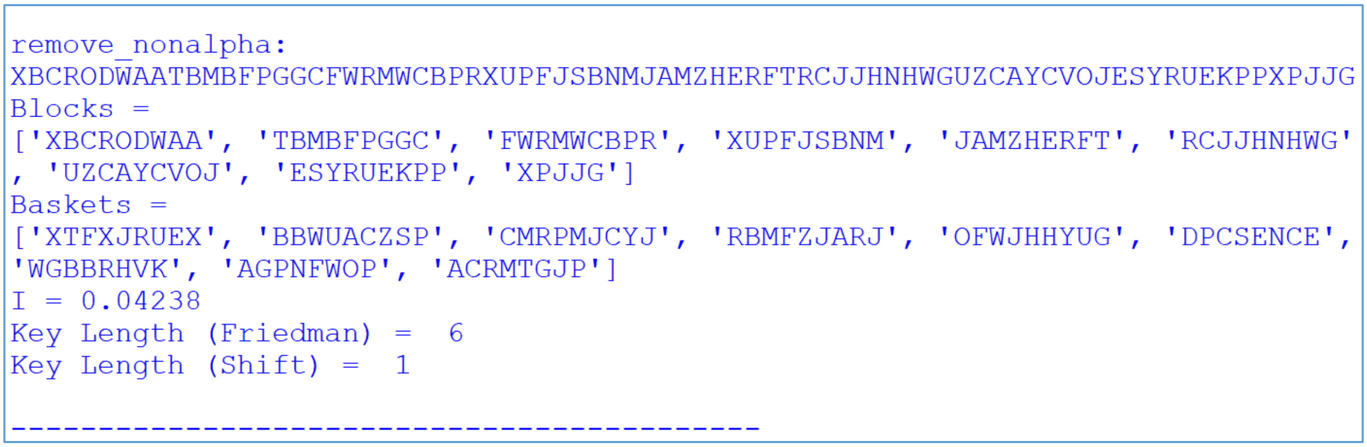
Note: Since the given ciphertext is short, it is no surprise that the key estimation tools produced inaccurate results.



```
>>> test_q2()
------------------------------------------
Testing Q2: Vigenere Cryptanalysis Utilities

remove_nonalpha:
HNTFUHMARDNDPLTWGTPIIACGRPIHGGTPWRNDTMDNNIHFKBOAXNRXWXELTOEGLOAOEPFAIAPGHBAXM
Blocks =
['HNT', 'FUH', 'MAR', 'DND', 'PLT', 'WGT', 'PII', 'ACG', 'RPI', 'HGG', 'TPW',
'RND', 'TMD', 'NNI', 'HFK', 'BOA', 'XNR', 'XWX', 'ELT', 'OEG', 'LOA', 'OEP',
'FAI', 'APG', 'HBA', 'XM']
Baskets =
['HFMDPWPARHTRTNHBXXEOLOFAHX', 'NUANLGICPGPNMNFONWLEOEAPBM', 'THRDTTIGIGWDDIK
ARXTGAPIGA']
I = 0.04819
Key Length (Friedman) =  3
Key Length (Shift) =  5

remove_nonalpha:
QECBNGVRAZGCYCCSZSYZRWVFAGRDZFCGFNGCCDMJGHQWTXHZGEATPWNCCKXFUFJKXOORRWIFQSJTF
Blocks =
['QECBNG', 'VRAZGC', 'YCCSZS', 'YZRWVF', 'AGRDZF', 'CGFNGC', 'CDMJGH', 'QWTXH
Z', 'GEATPW', 'NCCKXF', 'UFJKXO', 'ORRWIF', 'QSJTF']
Baskets =
['QVYYACCQGNUOQ', 'ERCZGGDWECFRS', 'CACRRFMTACJRJ', 'BZSWDNJXTKKWT', 'NGZVZGG
HPXXIF', 'GCSFFCHZWFOF']
I = 0.04511
Key Length (Friedman) =  4
Key Length (Shift) =  6
```



```
remove_nonalpha:
XBCRODWAATBMBFPGGCFWRMWCBPRXUPFJSBNMJAMZHERFTRCJJHNHWGUZCAYCVOJESYRUEKPPXPJJG
Blocks =
['XBCRODWAA', 'TBMBFPGGC', 'FWRMWCBPR', 'XUPFJSBNM', 'JAMZHERFT', 'RCJJHNHWG'
, 'UZCAYCVOJ', 'ESYRUEKPP', 'XPJJG']
Baskets =
['XTFXJRUEX', 'BBWUACZSP', 'CMRPMJCYJ', 'RBMFZJARJ', 'OFWJHHYUG', 'DPCSENCE',
'WGBBRHVK', 'AGPNFWOP', 'ACRMTGJP']
I = 0.04238
Key Length (Friedman) =  6
Key Length (Shift) =  1

------------------------------------------
```