User Guide

To use tihs program:

1. Change ciphertext in ciphertext.txt to be your texts
2. Under Change Directory and Import Modules, change directory path to where you store this program(may be needed depending on your machine configuration)
3. Increase the number of trials second keyword shift, the number of initial trials for the first substitution keyword, the number of swaps may increase accurary.

Encryption and decryption details can be found under the git repo. There is a pdf file.

Change Directory and Import Modules

In [6]:

```
# change directory to current folder, change this as you run it
#%cd /content/drive/My\ Drive/jupyter/decoder
import decrypt
from split_words import wordninja
from decrypt import vigenere_decryption as vdcp
from decrypt import mono_decryption as mdcp
import re
######## For notebook display only #############
# to disable output wrapping in notebook
from IPython.display import HTML, display
def set_css():
  display(HTML('''
  <style>
    pre {
        white-space: pre;
    }
  </style>
  '''))
get_ipython().events.register('pre_run_cell', set_css)
# to add outputwrap for the plaintext
import textwrap
wrapper = textwrap.TextWrapper(width=80,
    initial_indent=" " * 4,
    subsequent_indent=" " * 4,
    break_long_words=False,
    break_on_hyphens=False)
```

Handling ciphertext

In [7]:

```
f = open("ciphertext.txt", "r")          # ciphertext.txt contains the cipher-text, change the text as you use this program
msg = f.read()
msg = ''.join(filter(str.isalpha, msg))
msg = msg.upper()                      # now msg is a string that consists of only upper case letters
print("ciphertext:\n", msg)
```

```
ciphertext:
 UUJOTTPFLKTCYOUHNABTOKDFUTPDXXUXDZQUPHBSRIWGMEEPJQOHNGYEHCYGJIACAKPYWAHCOBUJURQOUIXIZOKDIZOVTFZRCRXKHOPJPHOPJAXLQOTUSKCKHLKCHUSJODJVNCOHUAMARRXPQXFDJNNUKEWEUREUJICZRTSXIZRMCJPHFKBBUYGPXTWRDAHCJOUJIJOUUYGOXBYCIBCPDZQXVD
MDHVPIOGHZPXQNCZZRVNJNESGEAVHRQMEPYMAOMQMUJIIPNICCHTTYFPXHJDOKRVNLNKWDSATJEGHHIRDNTFNPXQUNWKHYCITHSXCKKMGJNNLQOTTJGBMVINJMAHAYARYCIBCPDZCULQBXEHOBUUYGOXBLCHUAUACHHXDDMVODMU
```

Calculating Index of Coincidence and Finding the Second Keyword Length

In [8]:

```
second_keyword_length = vdcp().second_key_word_length(msg)
```

```
Suppose keyword length is 4. Frequency of letters:
```

```
A 4   B 0   C 5   D 4   E 3   F 2   G 1   H 7   I 7   J 8   K 2   L 2   M 4   N 6   O 6   P 3   Q 3   R 4   S 3   T 3   U 10   V 2   W 2   X 4   Y 2   Z 1
IC4 =  0.04334104775930991
Suppose keyword length is 5. Frequency of letters:
A 2   B 2   C 2   D 0   E 6   F 0   G 1   H 14   I 1   J 6   K 3   L 0   M 0   N 2   O 5   P 0   Q 2   R 7   S 0   T 9   U 10   V 3   W 0   X 3   Y 0   Z 0
IC5 =  0.08225108225108224
IC5 > IC4, second keyword length is 5
```

Frequency Strips and Finding Promising Second Keyword Shifts

In [9]:

```python
vdcp().visualize_strips(msg, second_keyword_length)
```

```
|  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  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|  2  |  2  |  2  |  0  |  6  |  0  |  1  |  1  |  1  |  6  |  3  |  0  |  0  |  2  |  5  |  0  |  2  |  7  |  0  |  9  |  1  |  3  |  0  |  3  |  0  |  0  |
|     |     |     |     |     |     |     |  4  |     |     |     |     |     |     |     |     |     |     |     |     |  0  |     |     |     |     |     |

|  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  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|  0  |  0  |  5  |  0  |  1  |  3  |  0  |  6  |  8  |  3  |  2  |  5  |  3  |  2  |  3  |  7  |  0  |  1  |  5  |  0  |  7  |  6  |  2  |  2  |  7  |  0  |

|  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  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|  5  |  0  |  9  |  1  |  1  |  2  |  6  |  1  |  1  |  4  |  4  |  0  |  0  |  7  |  2  |  4  |  4  |  6  |  0  |  1  |  0  |  0  |  1  |  4  |  3  |  1  |
|     |     |     |  2  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

|  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  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|  0  |  7  |  6  |  3  |  3  |  2  |  3  |  2  |  7  |  8  |  0  |  2  |  5  |  0  |  1  |  6  |  2  |  0  |  1  |  0  |  0  |  0  |  2  |  2  |  2  |  5  |
|     |     |     |     |     |     |     |     |     |     |     |     |     |     |  0  |     |     |     |     |     |     |     |     |     |     |     |

|  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  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|  9  |  3  |  1  |  2  |  0  |  0  |  1  |  3  |  0  |  0  |  6  |  0  |  5  |  6  |  3  |  2  |  4  |  1  |  1  |  6  |  1  |  0  |  1  |  8  |  1  |  5  |
|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |  0  |     |     |     |     |     |
```

Now, we let the program perform aligning frequency strips and return a few promising shifts.

In [10]:

```python
freq_strips = vdcp().frequency_strip(msg, second_keyword_length)
txt_len = len(msg)
guess_shifts = vdcp().get_guess_shifts(freq_strips, txt_len, 3)  # we return 3 most promising second keyword shifts
print(guess_shifts)
```

```
[(1, 22, 21, 6), (1, 3, 21, 6), (1, 22, 21, 19)]
```

Monoalphabetic Decoding With Guess Second Keyword Shifts

With some guess shifts, we try out each one of them.

In [11]:

```python
freq_dict = mdcp().get_frequency_dictionary()  # a hashmap, key: trigrams; value: relative frequency in Englsih
best_rating = 0        # initialize a rating for each key, updated key if we get better ratings
```

Loop through each guess second keyword shifts and do mono-alphabetic decoding on each one of them, and select the best result.

In [12]:

```python
for guess_shift in guess_shifts:
    second_keyword = "A"
    for s in guess_shift:
        second_keyword += chr(65+s)
    ciphertext = vdcp().vigenere_decrypt(msg, second_keyword, len(second_keyword))
    sub, rating = mdcp().get_key(ciphertext, 50, 5000, freq_dict)
    if rating > best_rating:
        best_rating = rating
        best_sub = sub
        best_v_keyword = second_keyword
```

In [13]:

```
print("The best second keyword shift found is:")
print(best_v_keyword)
print("Remember that this is not the second keyword, this is just the relative position. If you allign letter strips such that\n \
these letters are in one column, then one of the other columns is the second keyword.")
print("\nThe best substitution the program found is: ")
print("Plain Alphabet:      ",  "".join(mdcp().alpha))
print("Encryption Alphabet: ",  "".join(best_sub))
```

```
The best second keyword shift found is:
ABWVG
Remember that this is not the second keyword, this is just the relative position. If you allign letter strips such that
 these letters are in one column, then one of the other columns is the second keyword.

The best substitution the program found is: 
Plain Alphabet:       ABCDEFGHIJKLMNOPQRSTUVWXYZ
Encryption Alphabet:  TSXBHMQJUZDINRGVYEKOCWAPLF
```

Getting the Plaintext

In [14]:

```
ciphertext = vdcp().vigenere_decrypt(msg, best_v_keyword, len(best_v_keyword)) # first we do Vigenere decipher
plaintext =  mdcp().cipher_to_plain_with_key(ciphertext, best_sub)             # then monoalphabetic decipher
print("The decoded plain text is:")
print(plaintext)
```

```
The decoded plain text is:
IAMAMATHGRADUATEFROMTHESTATEUNIVERSITYOFNEWYORKATSTONYBROOKWHEREISTUDIEDBOTHAPPLIEDMATHEMATICSANDPUREMATHEMATICSIAMINTERESTEDINMACHINELEARNINGDISCRETEMATHGRAPHTHEORYANDMANYOTHERTOPICSINAPPLIEDMATHEMATICSANDCOMPUTERSCIEN
CEIAMLOOKINGFORANINTERNSHIPOPPORTUNITYINTHEFIELDOFMACHINELEARNINGESPECIALLYDEEPLEARNINGANDRECOMMENDERSYSTEMSIAMALSOOPENTOWORKINCOMPUTERVISIONROBOTICSANDSOFTWAREDEVELOPMENT
```

Word Segmentation

We may apply word segmentation to make the plaintext easier to read. This part uses an open source code wordninja. The segmentation is not perfect, but it is good enough for people to comprehend.

In [15]:

```
segmented_plaintext = wordninja.split(plaintext)
print("The segmented plaintext is:\n")
print(wrapper.fill(" ".join(segmented_plaintext).lower()))
```

```
The segmented plaintext is:

    iam a math graduate from the state university of new york at stony brook
    where i studied both applied mathematics and pure mathematics iam interested
    in machine learning discrete math graph theory and many other topics in
    applied mathematics and computer science iam looking for an internship
    opportunity in the field of machine learning especially deep learning and
    recommend er system siam also open to work in computer vision robotics and
    software development
```

Other Useful Functions

Vigenere Decryption by the Second Keyword

With known second keyword(performed frequency strips alignment by real human or just know the keyword), we may decode the msg to a ciphertext for the monoalphabetic decoder to decode.

In [16]:

```
second_keyword = "STONY"
ciphertext = vdcp().vigenere_decrypt(msg, second_keyword, len(second_keyword))
print("New ciphertext is:\n", ciphertext)
```

```
New ciphertext is:
 CBVBVBWRYMBJKBWPUMOVWRPSWBWPKZCEPMSCWTOUZPITOMLBWSWOZTAMOOLIRPMPCSWKJCPJAOWRBDDQCPJVBWRPVBWCFSBZJDKMPVBWRPVBWCFSCBVCZWPMPSWPJCZVBFRCZPQPBMZCZYJCSFMPWPVBWRYMBDRWRPOMTBZJVBZTOWRPMWODCFSCZBDDQCPJVBWRPVBWCFSBZJFOVDKWPMSFCP
ZFPCBVQOOLCZYUOMBZCZWPMZSRCDODDOMWKZCWTCZWRPUCPQJOUVBFRCZPQPBMZCZYPSDPFCBQQTJPPDQPBMZCZYBZJMPFOVVPZJPMSTSWPVSCBVBQSOODPZWOIOMLCZFOVDKWPMECSCOZMOAOWCFSBZJSOUWIBMPJPEPQODVPZW
```

Trigraph Table

Each trigram in the text is represented as the letter in the middle and its neighbors. For example, GE under A represents the trigram GAE.

In [17]:

```python
print("Trigraph Table:")
mdcp().trigraph_table(ciphertext)
```

```
Trigraph Table:

| 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   |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 3   | 35  | 35  | 16  | 3   | 14  | 0   | 0   | 4   | 16  | 7   | 4   | 25  | 0   | 28  | 46  | 11  | 15  | 20  | 8   | 6   | 22  | 34  | 0   | 6   | 32  |
| TM  | CV  | B   | BD  | CP  | CS  | -   | -   | PT  | BK  | JB  | MB  | YB  | -   | MV  | WU  | DC  | WY  | PW  | WO  | PM  | BB  | BR  | -   | RM  | KC  |
| JO  | VV  | ZE  | DQ  | MC  | CS  | -   | -   | LR  | KC  | PZ  | OI  | UO  | -   | TU  | RS  | PP  | WP  | MC  | IO  | OZ  | BB  | BP  | -   | ZJ  | UP  |
| OO  | VW  | SW  | JK  | PP  | BR  | -   | -   | OO  | PA  | WJ  | OC  | PS  | -   | TM  | WK  | DC  | IP  | WW  | ZA  | YO  | OW  | VR  | -   | RM  | OT  |
| -   | MJ  | PS  | BR  | -   | SM  | -   | -   | WB  | PV  | DM  | MC  | OL  | -   | WZ  | EM  | VO  | WB  | CW  | MB  | PC  | JB  | SB  | -   | ZU  | BJ  |
| -   | KW  | JP  | OC  | -   | CS  | -   | -   | -   | ZD  | DW  | -   | AO  | -   | MO  | ZI  | PJ  | WP  | FB  | ZO  | OV  | PB  | BP  | -   | ZP  | CW  |
| -   | WW  | QP  | BD  | -   | CS  | -   | -   | -   | PC  | WZ  | -   | PP  | -   | OL  | RM  | PP  | WP  | FC  | WC  | OW  | PB  | CT  | -   | ZB  | CV  |
| -   | LW  | WF  | DQ  | -   | JO  | -   | -   | -   | YC  | DW  | -   | KP  | -   | AW  | MC  | BQ  | FC  | PW  | QJ  | -   | PB  | BS  | -   | -   | CP  |
| -   | RD  | WF  | VK  | -   | SC  | -   | -   | -   | ZV  | -   | -   | PP  | -   | PM  | CJ  | QT  | WY  | CF  | SS  | -   | BC  | SO  | -   | -   | MC  |
| -   | VW  | SB  | CO  | -   | ZP  | -   | -   | -   | PV  | -   | -   | BZ  | -   | TW  | CJ  | DP  | DW  | FC  | -   | -   | ZB  | SK  | -   | -   | CY  |
| -   | VW  | VZ  | OD  | -   | BR  | -   | -   | -   | ZF  | -   | -   | FP  | -   | WD  | RV  | BS  | WP  | FB  | -   | -   | PB  | OR  | -   | -   | BJ  |
| -   | SZ  | JZ  | DO  | -   | PC  | -   | -   | -   | QO  | -   | -   | YB  | -   | FV  | MV  | PO  | WP  | MF  | -   | -   | JB  | BR  | -   | -   | BT  |
| -   | VW  | RZ  | SP  | -   | PO  | -   | -   | -   | TP  | -   | -   | OT  | -   | QO  | RV  | -   | WP  | ZR  | -   | -   | JB  | BC  | -   | -   | CB  |
| -   | VW  | ZZ  | PQ  | -   | ZO  | -   | -   | -   | ZM  | -   | -   | PW  | -   | OL  | WM  | -   | SC  | PD  | -   | -   | PB  | BR  | -   | -   | BJ  |
| -   | CV  | JS  | OP  | -   | CS  | -   | -   | -   | ZP  | -   | -   | PS  | -   | UM  | MS  | -   | WP  | MT  | -   | -   | OD  | BC  | -   | -   | PF  |
| -   | VF  | DF  | VK  | -   | -   | -   | -   | -   | ZS  | -   | -   | OB  | -   | DD  | WJ  | -   | FC  | TW  | -   | -   | BQ  | ZP  | -   | -   | CY  |
| -   | PM  | SZ  | OV  | -   | -   | -   | -   | -   | PP  | -   | -   | PZ  | -   | DM  | ZQ  | -   | -   | VC  | -   | -   | UB  | SP  | -   | -   | BC  |
| -   | VW  | QP  | -   | -   | -   | -   | -   | -   | -   | -   | -   | OW  | -   | JU  | QB  | -   | -   | QO  | -   | -   | OV  | PP  | -   | -   | CW  |
| -   | MD  | WF  | -   | -   | -   | -   | -   | -   | -   | -   | -   | BZ  | -   | FV  | MW  | -   | -   | CC  | -   | -   | VP  | BR  | -   | -   | MS  |
| -   | TZ  | FP  | -   | -   | -   | -   | -   | -   | -   | -   | -   | BZ  | -   | SO  | WV  | -   | -   | FB  | -   | -   | PS  | RR  | -   | -   | KC  |
| -   | VZ  | PB  | -   | -   | -   | -   | -   | -   | -   | -   | -   | JP  | -   | OD  | RO  | -   | -   | JO  | -   | -   | BB  | OR  | -   | -   | CW  |
| -   | ZD  | LZ  | -   | -   | -   | -   | -   | -   | -   | -   | -   | PS  | -   | WI  | RM  | -   | -   | -   | -   | -   | OD  | MO  | -   | -   | CP  |
| -   | VW  | ZZ  | -   | -   | -   | -   | -   | -   | -   | -   | -   | OL  | -   | IM  | CJ  | -   | -   | -   | -   | -   | DP  | BR  | -   | -   | MC  |
| -   | VW  | RD  | -   | -   | -   | -   | -   | -   | -   | -   | -   | PE  | -   | FV  | RV  | -   | -   | -   | -   | -   | -   | BC  | -   | -   | CY  |
| -   | SZ  | ZW  | -   | -   | -   | -   | -   | -   | -   | -   | -   | ZO  | -   | CZ  | WM  | -   | -   | -   | -   | -   | -   | KP  | -   | -   | MC  |
| -   | CV  | TZ  | -   | -   | -   | -   | -   | -   | -   | -   | -   | BP  | -   | MA  | CZ  | -   | -   | -   | -   | -   | -   | ZP  | -   | -   | CY  |
| -   | MZ  | UP  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | AW  | FC  | -   | -   | -   | -   | -   | -   | MK  | -   | -   | BJ  |
| -   | VF  | RZ  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | SU  | WM  | -   | -   | -   | -   | -   | -   | CT  | -   | -   | PJ  |
| -   | PM  | ZZ  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | QD  | RU  | -   | -   | -   | -   | -   | -   | ZR  | -   | -   | PW  |
| -   | CQ  | FB  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | CQ  | -   | -   | -   | -   | -   | -   | SP  | -   | -   | CF  |
| -   | PM  | ZZ  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | ZQ  | -   | -   | -   | -   | -   | -   | ZO  | -   | -   | OM  |
| -   | YZ  | SB  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | QB  | -   | -   | -   | -   | -   | -   | KP  | -   | -   | BJ  |
| -   | CV  | LZ  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | YS  | -   | -   | -   | -   | -   | -   | OC  | -   | -   | PW  |
| -   | VQ  | ES  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | DF  | -   | -   | -   | -   | -   | -   | UI  | -   | -   | -   |
| -   | SZ  | SO  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | JP  | -   | -   | -   | -   | -   | -   | Z   | -   | -   | -   |
| -   | IM  | WF  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | PD  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | QB  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | MF  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | VZ  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | JM  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | WV  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | DZ  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | WM  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | MJ  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | JE  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | EQ  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | VZ  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   |
```

Monoalphabetic Decryption by English Trigram Frequencies

The program attempts to decrypt the ciphertext by greedy. This may take a few minutes. Sometimes it is off by just one or two letters, a real human can fix it quickly.

In [ ]:

```python
# get_key takes 3 arguments: ciphertext, initial_trials, swaptrials; increase trial numbers may increase accurary
key = mdcp().get_key(ciphertext, 50, 2500)
```

In [ ]:

```
print("The best key that the program generated is: \n")
print("Plain Alphabet:      ",  "".join(mdcp().alpha))
print("Encryption Alphabet: ",  "".join(key))
```

```
The best key that the program generated is: 

Plain Alphabet:       ABCDEFGHIJKLMNOPQRSTUVWXYZ
Encryption Alphabet:  AZERTYUIOPQSDFGHJKLMWXCVBN
```

In [ ]:

```
plaintext =  mdcp().cipher_to_plain_with_key(ciphertext, key)
print("The plaintext decrypted with above key is:\n")
print(plaintext)
```

```
The plaintext decrypted with above key is:

IAMAMATHGRADUATEFROMTHESTATEUNIVERSITYOFNEWYORKATSTONYBROOKWHEREISTUDIEDBOTHAPPLIEDMATHEMATICSANDPUREMATHEMATICSIAMINTERESTEDINMACHINELEARNINGDISCRETEMATHGRAPHTHEORYANDMANYOTHERTOPICSINAPPLIEDMATHEMATICSANDCOMPUTERSCIEN
CEIAMLOOKINGFORANINTERNSHIPOPPORTUNITYINTHEFIELDOFMACHINELEARNINGESPECIALLYDEEPLEARNINGANDRECOMMENDERSYSTEMSIAMALSOOPENTOWORKINCOMPUTERVISIONROBOTICSANDSOFTWAREDEVELOPMENT
```