

## Indi 2 – Pac109

!!! This is an INDIVIDUAL project. Submit your project report as one file (word or pdf) in Canvas by 5pm, May 5. !!!

### ===== Project Description =====

The project is to crack a cipher text, which was encrypted using the Vigenere Cipher. The cipher is described in the lecture. You can also find many tutorials and examples online. A subtle difference is that the cipher uses 27 characters (instead of 26) in clear text, cipher text and key. The encryption  $E$  is  $c=(m+k)\%27$ . The decryption  $D$  is  $m=(27+c-k)\%27$ .

The characters include the upper case 'A' to 'Z' and the space ' '. 'A' is equivalent to 0. 'B' is equivalent to 1. ... 'Z' is equivalent to 25. And, the space ' ' is equivalent to 26.

For example, if the clear text is "HELLO WORLD" and the key is "ABC", then the cipher text is "HFNLPBWPTLE".

The cipher text of the project is in cipher.txt. The cipher text is only ONE line of characters. Please follow the cracking approach in the lecture to find the key. The key length is smaller than 10.

You need to develop a program(s) in C/C++, Python, Java, Matlab, or R to automate the crypto-analysis. If you

Manual analysis (such as using Excel) will not be counted towards your project grade.

### ===== Report and Grading =====

The score of the project is 10 points.

Your report should include what you did and the results you obtained in the four steps below.

Step 1: You will get 2 points if you are able to do the analysis on repeating patterns in the cipher text and do factorizing to guess the possible size of keys. Show a screenshot of the repeating patterns when running your analysis program.

```

Number of Characters: 5

Chars: FXGT Frequency: 2 Factors: [1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 25, 30, 40, 50, 60, 75, 100, 120, 150, 200, 300, 600] Distance: 600
Chars: FXGTL Frequency: 3 Factors: [1, 5, 17, 25, 85, 425] Distance: 425
Chars: XGTL Frequency: 3 Factors: [1, 5, 17, 25, 85, 425] Distance: 425
Chars: GTL N Frequency: 3 Factors: [1, 5, 17, 25, 85, 425] Distance: 425
Chars: SWNGX Frequency: 2 Factors: [1, 2, 5, 10, 25, 50, 125, 250] Distance: 250
Chars: WNGXL Frequency: 2 Factors: [1, 2, 5, 10, 25, 50, 125, 250] Distance: 250
Chars: NGXLP Frequency: 2 Factors: [1, 2, 5, 10, 25, 50, 125, 250] Distance: 250
Chars: GXLPLZ Frequency: 2 Factors: [1, 2, 5, 10, 25, 50, 125, 250] Distance: 250
Chars: HPOIF Frequency: 2 Factors: [1, 2, 4, 5, 8, 10, 16, 20, 32, 40, 64, 80, 160, 320] Distance: 320
Chars: GXRAG Frequency: 2 Factors: [1, 3, 5, 15, 19, 57, 95, 285] Distance: 285
Chars: XRAGY Frequency: 2 Factors: [1, 3, 5, 15, 19, 57, 95, 285] Distance: 285
Chars: RAGYE Frequency: 2 Factors: [1, 3, 5, 15, 19, 57, 95, 285] Distance: 285
Chars: AGYEZ Frequency: 2 Factors: [1, 3, 5, 15, 19, 57, 95, 285] Distance: 285
Chars: GYEZB Frequency: 2 Factors: [1, 3, 5, 15, 19, 57, 95, 285] Distance: 285
Chars: YEZBE Frequency: 2 Factors: [1, 3, 5, 15, 19, 57, 95, 285] Distance: 285
Chars: EZBEZ Frequency: 2 Factors: [1, 3, 5, 15, 19, 57, 95, 285] Distance: 285
Chars: ZBEZW Frequency: 2 Factors: [1, 3, 5, 15, 19, 57, 95, 285] Distance: 285
Chars: KGPNZ Frequency: 2 Factors: [1, 3, 5, 7, 15, 21, 35, 105] Distance: 105
Chars: SWNGX Frequency: 2 Factors: [1, 2, 5, 10, 25, 50, 125, 250] Distance: 250
Chars: WNGXL Frequency: 2 Factors: [1, 2, 5, 10, 25, 50, 125, 250] Distance: 250
Chars: NGXLP Frequency: 2 Factors: [1, 2, 5, 10, 25, 50, 125, 250] Distance: 250
Chars: GXLPLZ Frequency: 2 Factors: [1, 2, 5, 10, 25, 50, 125, 250] Distance: 250
Chars: LPZGF Frequency: 2 Factors: [1, 3, 5, 15, 17, 51, 85, 255] Distance: 255
Chars: KGPNZ Frequency: 2 Factors: [1, 3, 5, 7, 15, 21, 35, 105] Distance: 105
Chars: NKULO Frequency: 2 Factors: [1, 2, 4, 5, 8, 10, 20, 40] Distance: 40
Chars: KULOA Frequency: 2 Factors: [1, 2, 4, 5, 8, 10, 20, 40] Distance: 40
Chars: ULOAX Frequency: 2 Factors: [1, 3, 5, 13, 15, 39, 65, 195] Distance: 195
Chars: LOAXZ Frequency: 2 Factors: [1, 3, 5, 13, 15, 39, 65, 195] Distance: 195
Chars: GNAVJ Frequency: 2 Factors: [1, 2, 5, 10, 19, 38, 95, 190] Distance: 190
Chars: GFDEW Frequency: 2 Factors: [1, 2, 4, 5, 8, 10, 20, 25, 40, 50, 100, 200] Distance: 200
Chars: FDEWY Frequency: 2 Factors: [1, 2, 4, 5, 8, 10, 20, 25, 40, 50, 100, 200] Distance: 200
Chars: DEWYZ Frequency: 2 Factors: [1, 2, 4, 5, 8, 10, 20, 25, 40, 50, 100, 200] Distance: 200
Chars: EWYZV Frequency: 2 Factors: [1, 2, 4, 5, 8, 10, 20, 25, 40, 50, 100, 200] Distance: 200
Chars: WYZVA Frequency: 2 Factors: [1, 2, 4, 5, 8, 10, 20, 25, 40, 50, 100, 200] Distance: 200

```

|    |   |  |
|----|---|--|
| 1  | Matches# 0: DSNJVHXSPPHSMGUSLVGLVLTZUI---Match Count: 26                                    |  |
| 2  | Matches# 1: UYNGVZ#IAZUCTUGKZGFSUGGZZT---Match Count: 26                                    |  |
| 3  | Matches# 2: ZUVNGYVFTVHXEZHYHGZGEZWGDDZGZKFXGZZJIWX---Match Count: 41                       |  |
| 4  | Matches# 3: XOAGOGGZLGAGLAGFLGLCRGULOULWT---Match Count: 29                                 |  |
| 5  | Matches# 4: EHPXHKGWPFGGKLNKGTGNJLGGXJANKWLJFFTXGZLJLWS#DUV---Match Count: 47               |  |
| 6  | Matches# 5: WXT#XYHPKHALDGLTGHFL#TZI---Match Count: 24                                      |  |
| 7  | Matches# 6: IZKGXNTGLNCZOGZBFLAGW---Match Count: 21   |  |
| 8  | Matches# 7: ZUHGKAKXRGBUGGFULZJULU---Match Count: 22  |  |
| 9  | Matches# 8: #KZXLWLNNGNSHOUNUGVVKXGJFLYZIIUUS---Match Count: 32                             |  |
| 10 | Matches# 9: LTZZTXNXZVXXSSCHWUSSGNKULZFCZFGZGFFFFGUSLWSVSEW---Match Count: 48               |  |
| 11 | Matches# 10: VLAXSHWHLUHSPLAUGXGJAX#FG#POLZLO---Match Count: 33                             |  |
| 12 | Matches# 11: TNNGNXINYUAUKZNGUEAGGNTMFDZXFLX---Match Count: 32                              |  |
| 13 | Matches# 12: ZWTENGZAZVSUGKKZNUUPGUVVXFCOSUWT---Match Count: 33                             |  |
| 14 | Matches# 13: ZNXNVAGIJSWNLXVJVZZLXTQU---Match Count: 24                                     |  |
| 15 | Matches# 14: XXNGPZFGTZXZHKNAXPXGCGZCGGWFLFWWI#S#VQ---Match Count: 38                       |  |
| 16 | Matches# 15: TZHVHMNLETFGJAXLKZGXULZ---Match Count: 25                                      |  |
| 17 | Matches# 16: ZXWXHYEEZKHHYXHYNWZALKKXGCGWZ#CUEOSDUUWT---Match Count: 41                     |  |
| 18 | Matches# 17: EGLYIGYUKKSEXT#AGNKVFFIDU---Match Count: 26                                    |  |
| 19 | Matches# 18: WZKXKLZAAZSHGGUHLUALVZGFCLTULLM---Match Count: 34                              |  |
| 20 | Matches# 19: SUNZUMXYUEEXFGSSHHYGUWSXSLVZDKAYG#XUGLLWFLFAFTLFLS#A#LA#IVSF---Match Count: 62 |  |
| 21 | Matches# 20: SSXZGNJXSLCUSWLUALWGAZAKZLLSUI---Match Count: 29                               |  |
| 22 | Matches# 21: STUJYUHWGXCZFGYXZWLFXDYDU---Match Count: 26                                    |  |
| 23 | Matches# 22: TZXZHGAKZKUNGHKKAVGZTUXFQANFSIDEK---Match Count: 32                            |  |
| 24 | Matches# 23: XNXHSGSNLHAGLAGJLEJZFQNZOOLZVI---Match Count: 30                               |  |
| 25 | Matches# 24: KTXNNLUYIIVHGYSHSYLNALNUUXGUKGWLFXZXFFNZXUVI---Match Count: 45                 |  |
| 26 | Matches# 25: NZ#HSHSHJALFGLXLLGZFSU---Match Count: 22                                       |  |
| 27 | Matches# 26: EZNFVHGWGGLAJXGVGGLJAL#UXZSE---Match Count: 29                                 |  |
| 28 | Matches# 27: BTUNYOGVVH#VJHGKWWGPPWUZZXLOOX---Match Count: 31                               |  |
| 29 | Matches# 28: ZGSWXKNNSVNVLUGOJ#LFXGLGNLZQS---Match Count: 29                                |  |
| 30 | Matches# 29: GWUGGLFWGAVHOGXASSUWFHWHKSNLZKNKULNAECELUCLEFFFFZSLSVDSUFA---Match Count: 58   |  |
| 31 | Matches# 30: HUVVVAZSGWHUFKXLXG#VZFGYIVSLJX---Match Count: 30                               |  |
| 32 | Matches# 31: ZX#GV0IYZAALXUUXGHZGPCXALCJXCJZISO---Match Count: 35                           |  |
| 33 | Matches# 32: UGGJZHXAZZHGZLXELFXLGZF#GUZ---Match Count: 27                                  |  |
| 34 | Matches# 33: KNZVLIZJVKAJOOZGSCWLSXAULFEWKVAFGXSF---Match Count: 38                         |  |
| 35 | Matches# 34: TTZXGTYZJUYZJSGYFAFAGKWGGFGUFNVAGZZG#LUZOFLWZL##ZL---Match Count: 51           |  |
| 36 | Matches# 35: HWXMKHGHHLGKAUZOGXFJCLXGNSSU---Match Count: 28                                 |  |
| 37 | Matches# 36: ZWVNNZIEKWXZYHVNKFG#WBCVWGX---Match Count: 28                                  |  |
| 38 | Matches# 37: XY#XKYTONVZXGUSGGUAUCJFGZOFI---Match Count: 28                                 |  |
| 39 | Matches# 38: NPNIHGGSNXLSGYAFYFGGGLLLUS---Match Count: 26                                   |  |

So I went about solving this problem two different ways. The first screenshot shows the algorithm the professor used in lecture where it finds the chars that repeat and it finds the distance between the repeats and then factorizes them. I was able to see that 5 was the most likely but 2,3,4,6 also seemed likely. To help discern which factor to use first I found another algorithm online that suggested to find the number of matches per line and then increments whenever you shift the cypher text by one and whenever that's done you find which lines give the highest number to find a pattern between them. That number also happened to be 5 so I used that as my first potential key size.

Step 2: After step 1, you will get 2 more points, if you are able to divide the cipher text into groups based on the guessed key size, do the 1-gram character analysis for all groups, and then guess the possible key characters for the groups. Show a screenshot of the cipher text groups when running your analysis program.

```
(base) PS C:\Users\phili\Desktop\pythonindi2> python analysis.py
EFHTATSMEEETXTGASLVG DLTXXGVLFTKAEZ GEJSCHAQSJSKGVXSALXMSXLAUYGKSCSLSKK SLYLHIAADVLDXTSEKXAVRFGEWLGAXDJJXGGGVFQHLSLGA LDVXKTGQFXCNAFLGAGTTS XXSLSEJJYDSX
SSULXDJUMELDXSTK

VG WZBIGHHSVUTYLYPNYNOYZKAVHHYZAVHGGYZKIGYJGHZPGYHVIHJZTGZLSGVHUMPTGLLHOVLGAVEAO GULWTGUY HGYZGLTUJVLZSUGVAG OLGVLPEAJZ BGVGLL SLTUCUTPLGGIOZLLLM PT
LO SLS WPSJTU

ZKDWEKKPPWN LWJNZUNZKIAEWJIPH WIJGOXEWMNEXAX XPPAHKHMGDUWEWA WJKNZAPWINANNZPJWNAEXZCNAEPWKPXEWNLCDCMMAQOYJXXAATEIJKZWNDWXXZPYJJDE LCWLWPLQAZOWMEWIA
DDOAEUXAWQKUQAE

KXMDNRYTNOZDF YIGKVGXFYFMGJVOTGVYO RZNZKTIFTAXFNZRYKHFFYHZOYFUHDBGGGNYKKXSUGGYKUIJKFUTTKLXORZGGF FTD ZFZFKZRJXXOTGFXYKKZVHGFXXFFSF FXKRUFUOGZYNFJJIZVT
GKZUIFQFIZTFXIY

LSLUZSUZXBUXNXZJUNJVCUYHLUOIBZMNUHXAABVIVUNMSZSCZCSMULMVNOUHCNUSUHXFHZXPMUZGNFZULVPMCPNJAVCIABHMXNNIUCBMCNUCFVUZH NNZFZUGBLFWIZQOCXNJURZNHGYUZZVICOIFN
HUULVWVWIVVMZBG

1 Line 1: ('A', 12) ('B', 0) ('C', 3) ('D', 8) ('E', 9) ('F', 6) ('G', 15) ('H', 4) ('I', 0) ('J', 7) ('K', 9) ('L', 17) ('M', 2) ('N', 1)
2 Line 2: ('A', 6) ('B', 2) ('C', 1) ('D', 0) ('E', 2) ('F', 0) ('G', 22) ('H', 11) ('I', 4) ('J', 5) ('K', 2) ('L', 20) ('M', 1) ('N', 2)
3 Line 3: ('A', 17) ('B', 1) ('C', 4) ('D', 7) ('E', 13) ('F', 0) ('G', 2) ('H', 3) ('I', 6) ('J', 9) ('K', 8) ('L', 5) ('M', 1) ('N', 12)
4 Line 4: ('A', 1) ('B', 1) ('C', 0) ('D', 4) ('E', 0) ('F', 26) ('G', 16) ('H', 4) ('I', 7) ('J', 5) ('K', 15) ('L', 1) ('M', 2) ('N', 6)
5 Line 5: ('A', 3) ('B', 7) ('C', 11) ('D', 0) ('E', 0) ('F', 6) ('G', 4) ('H', 9) ('I', 8) ('J', 2) ('K', 0) ('L', 7) ('M', 9) ('N', 16)
6

('N', 1) ('O', 0) ('P', 0) ('Q', 3) ('R', 1) ('S', 20) ('T', 11) ('U', 3) ('V', 7) ('W', 3) ('X', 18) ('Y', 3) ('Z', 1) (' ', 5)
('N', 2) ('O', 6) ('P', 8) ('Q', 0) ('R', 0) ('S', 7) ('T', 9) ('U', 9) ('V', 12) ('W', 6) ('X', 0) ('Y', 10) ('Z', 11) (' ', 11)
('N', 12) ('O', 5) ('P', 11) ('Q', 4) ('R', 0) ('S', 0) ('T', 1) ('U', 4) ('V', 0) ('W', 25) ('X', 12) ('Y', 2) ('Z', 9) (' ', 6)
('N', 6) ('O', 7) ('P', 0) ('Q', 1) ('R', 6) ('S', 3) ('T', 10) ('U', 6) ('V', 5) ('W', 0) ('X', 10) ('Y', 12) ('Z', 14) (' ', 5)
('N', 16) ('O', 4) ('P', 2) ('Q', 1) ('R', 1) ('S', 6) ('T', 0) ('U', 24) ('V', 12) ('W', 4) ('X', 9) ('Y', 2) ('Z', 19) (' ', 1)
```

Step 3: After step 2, you will get 2 more points, if you are able to guess at least two correct key characters and crack the corresponding 2 groups of the cipher text. Show a screenshot of the key characters when running your analysis program.

I noticed a pattern repeating throughout the different 1 gram lines that being:

Line 1: ('S', 20) ('T', 11) ('U', 3) ('V', 7) ('W', 3) ('X', 18) – HMLLLLH

Line 2: ('G', 22) ('H', 11) ('I', 4) ('J', 5) ('K', 2) ('L', 20) – HMLLLLH

LINE 3: ('W', 25) ('X', 12) ('Y', 2) ('Z', 9) (' ', 6) ('A', 17) – HMLLLLH

LINE 4: ('F', 26) ('G', 16) ('H', 4) ('I', 7) ('J', 5) ('K', 15) – HMLLLLH

LINE 5: ('U', 24) ('V', 12) ('W', 4) ('X', 9) ('Y', 2) ('Z', 19) – HMLLLLH

Based on this information I decided to use the start of each patten as my key for the decryption.

```
key = "SGWFU"
```

And we get garbage:

```
Anaconda Powershell Prompt (anaconda3)
ATDIFNFTABSFAFPFAEJGGJDMUAPAJNQMFNFOUAIBOAUIFACFTUAUIFPSFUJDBMMZACSFBLBMCFAVUADPNQVUBUJPOBMMZATFDVVSFANFDIOBJTNT
(base) PS C:\Users\phili\Desktop\pythonindi2> python decrypt.py
['EV KL', 'FGZXS', 'H KML', 'TWDDU', 'AZWNZ', 'TBERS', 'SIXYZ', 'WGKTU', 'EHPNZ', 'EHPOX', 'TSWZB', 'XVNDU', 'TU FX', 'GTL N', 'XYWYX', 'ALJIZ',
'SWNGX', 'LPZKU', 'VYUVN', 'GNNGJ', 'PZFB', 'DNKXC', 'LOIYU', 'TYAFY', 'XZEMH', 'XKWL', 'GAJJU', 'WVIVO', 'LHPOI', 'FHHFB', 'TY TZ', 'KZWGM',
'KAIVN', 'AVJYU', 'EHGOH', 'ZGO X', 'GXRA', 'GYEZB', 'EZMNV', 'JKWZI', 'SINKV', 'CGETU', 'HYXIN', 'AJAFW', 'QGXTS', 'SH AZ', 'JZXS', 'SPPFC',
'KGNZ', 'GYAZC', 'VHHR', 'SWKYM', 'AIHKU', 'LVHHL', 'XHGFM', 'MJDV', 'SZUYN', 'XTWHO', 'LGEZU', 'AZWOH', 'YLXYC', 'USAFN', 'GG UU', 'KVMHS',
'SHJDU', 'CUKBH', 'SWNGX', 'LPZGF', 'STAGH', 'KGNZ', 'KLWYX', 'LIK', 'SHNKU', 'LOAXZ', 'VYVNU', 'LLNSZ', 'WGZUG', 'HAPGN', 'AVJGF', 'DEWYZ',
'VANKU', 'LOAUL', 'X EIV', 'DGXJP', 'TUZKM', 'SLCFC', 'EMNUP', 'XTATN', 'KGETU', 'AUPKA', 'XYWLV', 'V KXC', 'RHPOI', 'FGXRA', 'GYEZB', 'EZWGH',
'WGBGM', 'LLNFX', 'GTL N', 'AUCFN', 'XJDTI', 'DVCDU', 'JLM C', 'JLWZB', 'XZAFM', 'GSQZC', 'GUOFN', 'GGYKU', 'VJZC', 'FAXRF', 'QGZJV', 'H AJU',
'LOAXZ', 'SLTOM', 'LGET', 'GYIGN', 'AVJFN', 'LKXZ', 'LPZGF', 'DEWYZ', 'VANKU', 'KJDKG', 'XZWZB', 'T WVL', 'GBXHF', 'QGZGH', 'FVPFW', 'XGYXI',
'CLJFZ', 'NLJFQ', 'A DFO', 'FSESC', 'LL FX', 'GTL N', 'AUCFJ', 'GCAXU', 'TUMKR', 'TTLRZ', 'SPOFN', 'LWUH', 'XGPOG', 'XGLGY', 'SIQZU', 'LOAYZ',
'SZZNZ', 'ELOFV', 'JLWSI', 'JLWJC', 'YMEIO', 'D WZI', 'SPIVF', 'XTATN', 'S DGH', 'S DKU', 'ULOZU', 'LOAUL', 'X EIV', 'DSUFW', 'JLXQV', 'USAFW',
'M WII', 'EQQZV', 'LPKTV', 'DSUFW', 'XJQXZ', 'STAIB', 'TUEYG', 'K']

['NPEFS', 'OADSZ', 'QUPHS', 'BQIZA', 'JTAIF', 'BWJMJ', 'ACBTF', 'EAPOA', 'NBUIF', 'NBUJD', 'BMAUI', 'FPSZA', 'BOEAD', 'PNQVU', 'FSATD', 'JFODF',
'AQSD', 'UJFDA', 'DSZQU', 'PHSBO', 'IJDAB', 'MHP SJ', 'UINTA', 'BSFAE', 'FTJHO', 'FEABS', 'PVOEA', 'DPNQV', 'UBUJP', 'OBMAI', 'BSEOF', 'TTABT',
'TVNUQ', 'JPOTA', 'NBLJO', 'HATVD', 'IABMH', 'PSJUI', 'NTAIB', 'SEAU', 'ACSFB', 'LAJOA', 'QSDU', 'ZABOZ', 'ABEW', 'STBSZ', 'AJUAJ',
'TAUTI', 'PSFUJ', 'DBMMZ', 'AQPTT', 'JCMFA', 'UPACS', 'FBLAT', 'VDIAB', 'ATZTU', 'FNACV', 'UAJUA', 'JTAJO', 'GFBTJ', 'CMFAU', 'PAEPA', 'TPACZ',
'ABOZA', 'LOPKO', 'AQSD', 'UJDBM', 'ANFBO', 'TAUIF', 'TFATD', 'IFNFT', 'ABSFA', 'UIFSF', 'GPSFA', 'UFSNF', 'EADPN', 'QVUBU', 'JPOBM', 'MZATF',
'DVSFA', 'UJFAS', 'FUJDB', 'MABEW', 'BODFT', 'AFHAJ', 'NQSPW', 'FNFOU', 'TAJOA', 'JOUFH', 'FSAGB', 'DUPSJ', 'BUJP', 'OABMH', 'PSJUI', 'NTABO',
'EAGBT', 'UFSAD', 'PNQVQ', 'JOHAU', 'FDIOP', 'MPHZA', 'SFRVJ', 'SFAUT', 'FTFAT', 'PMVUJ', 'POTAU', 'PACFA', 'DPOUJ', 'OVBMH', 'ZABEB', 'QUFEA',
'UIFSF', 'AFYJT', 'UAJOG', 'PSNBU', 'IFPSF', 'UJDBM', 'MZATE', 'DVSFA', 'TDIFN', 'FTAUI', 'BUAQ', 'PWBCH', 'ZADBO', 'OPUAC', 'OPUAC', 'QUFEA',
'LOFAF', 'WFOAX', 'JUTAV', 'OMJNJ', 'UFEAD', 'PNQVU', 'JOHAQ', 'PXFS', 'BOAFY', 'BNQMF', 'AJTAU', 'IFAPO', 'FAUJN', 'FAQBE', 'ACVUA', 'UITFF',
'ATDIF', 'NFTAB', 'SFANP', 'SFAEJ', 'GGJDV', 'MUAUP', 'AJNQ', 'FNFOU', 'AUIBO', 'AUIFA', 'CFTUA', 'UIFPS', 'FUJDB', 'MMZAC', 'SFBBL', 'CMFAC',
'VUADP', 'NQVUB', 'UJPOB', 'MMZAT', 'FDVSF', 'ANFDI', 'BOJTN', 'T']
(base) PS C:\Users\phili\Desktop\pythonindi2>
```

I was still sure that the patten I had found was the key based on the lecture slides so I shifted everything over to the right one and got `key = "THXGV"` as the key.

Which gave the results:

```
Anaconda Powershell Prompt (anaconda3)
['VUADP', 'NQVUB', 'UJPOB', 'MMZAT', 'FDVSF', 'ANFDI', 'BOJTN', 'T']
(base) PS C:\Users\phili\Desktop\pythonindi2> python decrypt.py
['EV KL', 'FGZXS', 'H KML', 'TWDDU', 'AZWNZ', 'TBERS', 'SIXYZ', 'WGKTU', 'EHPNZ', 'EHPOX', 'TSWZB', 'XVNDU', 'TU FX', 'GTL N', 'XYWYX', 'ALJIZ',
'SWNGX', 'LPZKU', 'VYUVN', 'GNNGJ', 'PZFB', 'DNKXC', 'LOIYU', 'TYAFY', 'XZEMH', 'XKWL', 'GAJJU', 'WVIVO', 'LHPOI', 'FHHFB', 'TY TZ', 'KZWGM',
'KAIVN', 'AVJYU', 'EHGOH', 'ZGO X', 'GXRA', 'GYEZB', 'EZMNV', 'JKWZI', 'SINKV', 'CGETU', 'HYXIN', 'AJAFW', 'QGXTS', 'SH AZ', 'JZXS', 'SPPFC',
'KGNZ', 'GYAZC', 'VHHR', 'SWKYM', 'AIHKU', 'LVHHL', 'XHGFM', 'MJDV', 'SZUYN', 'XTWHO', 'LGEZU', 'AZWOH', 'YLXYC', 'USAFN', 'GG UU', 'KVMHS',
'SHJDU', 'CUKBH', 'SWNGX', 'LPZGF', 'STAGH', 'KGNZ', 'KLWYX', 'LIK', 'SHNKU', 'LOAXZ', 'VYVNU', 'LLNSZ', 'WGZUG', 'HAPGN', 'AVJGF', 'DEWYZ',
'VANKU', 'LOAUL', 'X EIV', 'DGXJP', 'TUZKM', 'SLCFC', 'EMNUP', 'XTATN', 'KGETU', 'AUPKA', 'XYWLV', 'V KXC', 'RHPOI', 'FGXRA', 'GYEZB', 'EZWGH',
'WGBGM', 'LLNFX', 'GTL N', 'AUCFN', 'XJDTI', 'DVCDU', 'JLM C', 'JLWZB', 'XZAFM', 'GSQZC', 'GUOFN', 'GGYKU', 'VJZC', 'FAXRF', 'QGZJV', 'H AJU',
'LOAXZ', 'SLTOM', 'LGET', 'GYIGN', 'AVJFN', 'LKXZ', 'LPZGF', 'DEWYZ', 'VANKU', 'KJDKG', 'XZWZB', 'T WVL', 'GBXHF', 'QGZGH', 'FVPFW', 'XGYXI',
'CLJFZ', 'NLJFQ', 'A DFO', 'FSESC', 'LL FX', 'GTL N', 'AUCFJ', 'GCAXU', 'TUMKR', 'TTLRZ', 'SPOFN', 'LWUH', 'XGPOG', 'XGLGY', 'SIQZU', 'LOAYZ',
'SZZNZ', 'ELOFV', 'JLWSI', 'JLWJC', 'YMEIO', 'D WZI', 'SPIVF', 'XTATN', 'S DGH', 'S DKU', 'ULOZU', 'LOAUL', 'X EIV', 'DSUFW', 'JLXQV', 'USAFW',
'M WII', 'EQQZV', 'LPKTV', 'DSUFW', 'XJQXZ', 'STAIB', 'TUEYG', 'K']

['MODER', 'N CRY', 'PTOGR', 'APHY', 'IS HE', 'AVILY', 'BASE', 'D ON', 'MATHE', 'MATIC', 'AL TH', 'EORY', 'AND C', 'OMPUT', 'ER SC', 'IENCE',
'PRAC', 'TICE', 'CRYPT', 'OGRAP', 'HIC A', 'LGORI', 'THMS', 'ARE D', 'ESIGN', 'ED AR', 'OUND', 'COMPU', 'TATIO', 'NAL H', 'ARDNE', 'SS AS',
'SUMPT', 'IONS', 'MAKIN', 'G SUC', 'H ALG', 'ORITH', 'MS HA', 'RD TO', 'BREA', 'K IN', 'PRACT', 'ICE B', 'Y ANY', 'ADVE', 'RSARY', 'IT I',
'S THE', 'ORETI', 'CALLY', 'POSS', 'IBLE', 'TO BR', 'EAK S', 'UCH A', 'SYST', 'EM BU', 'T IT', 'IS IN', 'FEASI', 'BLE T', 'O DO', 'SO BY',
'ANY', 'KNOWN', 'PRAC', 'TICAL', 'MEAN', 'S THE', 'SE SC', 'HEMES', 'ARE', 'THERE', 'FORE', 'TERME', 'D COM', 'PUTAT', 'IONAL', 'LY SE',
'CURE', 'THEOR', 'ETICA', 'L ADV', 'ANCES', 'EG I', 'MPROV', 'EMENT', 'S IN', 'INTEG', 'ER FA', 'CTORI', 'ZATIO', 'N ALG', 'ORITH', 'MS AN',
'D FAS', 'TER C', 'OMPUT', 'ING T', 'ECHNO', 'LOGY', 'REQUI', 'RE TH', 'ESE S', 'OLUTI', 'ONS T', 'O BE', 'CONTI', 'NUALL', 'Y ADA', 'PTED',
'THERE', 'EXIS', 'T INF', 'ORMAT', 'ION T', 'HEORE', 'TICAL', 'LY SE', 'CURE', 'SCHEM', 'ES TH', 'AT PR', 'OVABL', 'Y CAN', 'NOT B', 'E BRO',
'KEN E', 'VEN W', 'ITH U', 'NLIMI', 'TED C', 'OMPUT', 'ING P', 'OWER', 'AN EX', 'AMPLE', 'IS T', 'HE ON', 'E TIM', 'E PAD', 'BUT', 'THESE',
'SCHE', 'MES A', 'RE MO', 'RE DI', 'FFICU', 'LT TO', 'IMPL', 'EMENT', 'THAN', 'THE', 'BEST', 'THEOR', 'ETICA', 'LLY B', 'REAKA', 'BLE B',
'UT CO', 'MPUTA', 'TIONA', 'LLY S', 'ECURE', 'MECH', 'ANISM', 'S']
(base) PS C:\Users\phili\Desktop\pythonindi2>
```

SUCCESS!

Step 4: After step 3, you will get 2 more points, if you are able to find the correct key and the clear text. Show a screenshot of the clear text when running your analysis program.

```
Anaconda Powershell Prompt (anaconda3)
(base) PS C:\Users\phili\Desktop\pythonind2> python decrypt.py
EV KLF6ZXS8 KMLTWDDUAZWNZTB
ERSSIXYZW8KTUEHPNZEHPXTSWZBXVNDUTU FXGTL NXYWYXALJZSWNGXLPZKUYVUVNGNNGJ PZFDN8XCLOIYUTYAFYXZEMH8KWGLGAJJUVVIVOLHPOIFHHFBTY TZKZWGMKAI8NAVJYUEHGOHZGO
X GXRAGYEZBEZWN8VKWZISINKVCGETUHYXINAJAFWQ8XTSS8 AZJZXSSPP8CKGPNZGYAZCVHHRSSWKYMAIHKULVWHLXHGFM8JDFVSZUYNXTWHOLGEZUAZWOHYLYXCUSAFNGG UUKVWHSS8JDUCUKB8S
WNGXLPZGFSTAGHKGPNZKLWYX LIKMSHNKULOAXZYVNKULLNSZW8ZUGHAP8NAVJGFDEWYZVANKULO8ULX EIVDGXJPTUZKMSLCFCEWNUPXTATN8KETUAUPKAXYWL8V KXCRHPOIFGXRAGYEZBEZ8GHW8B
GMLL8FXGTL NAUCFN8JDTIDVCDUJLM CJLWZBX8AFM8SQZCGUOFNGGYKUVVJZCFAXRFQGXJVH AJULOAXZSLTOMLGET GYIGNAVJFN LKXZLPZGFDEWYZVANKUKJDKGXZ8WBT WVLGBXHFQ8ZGHFVPFW
XGYICLJFZNLJFQA DFOFSE8CLL FXGTL NAUCFJGCAXUTUWKRTTLRZ8POFN LWUHXGPOGXGLGYSIQZULOAYZ8Z8NZELOFVJLWSIJLWJCYMEIOD WZISPIV8XTATNS DGHS DKUULOZULO8ULX EIVDS
UFWJLXQVUSAFWM WIEWQZVLPKTVDSUFMXJQXZ8TAIBTUEYGK

MODERN CRYPTOGRAPHY IS HEAVILY BASED ON MATHEMATICAL THEORY AND COMPUTER SCIENCE PRACTICE CRYPTOGRAPHIC ALGORITHMS ARE DESIGNED AROUND COMPUTATIONAL HAR
DNESS ASSUMPTIONS MAKING SUCH ALGORITHMS HARD TO BREAK IN PRACTICE BY ANY ADVERSARY IT IS THEORETICALLY POSSIBLE TO BREAK SUCH A SYSTEM BUT IT IS INFEAS
IBLE TO DO SO BY ANY KNOWN PRACTICAL MEANS THESE SCHEMES ARE THEREFORE TERMED COMPUTATIONALLY SECURE THEORETICAL ADVANCES EG IMPROVEMENTS IN INTEGER FAC
TORIZATION ALGORITHMS AND FASTER COMPUTING TECHNOLOGY REQUIRE THESE SOLUTIONS TO BE CONTINUALLY ADAPTED THERE EXIST INFORMATION THEORETICALLY SECURE SCH
EMES THAT PROBABLY CANNOT BE BROKEN EVEN WITH UNLIMITED COMPUTING POWER AN EXAMPLE IS THE ONE TIME PAD BUT THESE SCHEMES ARE MORE DIFFICULT TO IMPLEMENT
THAN THE BEST THEORETICALLY BREAKABLE BUT COMPUTATIONALLY SECURE MECHANISMS
(base) PS C:\Users\phili\Desktop\pythonind2>
```

Step 5: Attach the source code of your program at the end of your report to get 2 more points.

\*\*\*NOTE\*\*\*

My code is in python and is weird about indentations and since copy and paste is really bad about keeping the indentations where they need to be, I am including the source code as separate files in my submission.

```
import os

from collections import Counter

class analysis():

    cypher = "EV KLF6ZXS8 KMLTWDDUAZWNZTB8ERSSIXYZW8KTUEHPNZEHPXTSWZBXVNDUTU FXGTL
NXYWYXALJZSWNGXLPZKUYVUVNGNNGJ
PZFDN8XCLOIYUTYAFYXZEMH8KWGLGAJJUVVIVOLHPOIFHHFBTY TZKZWGMKAI8NAVJYUEHGOHZGO X
GXRAGYEZBEZWN8VKWZISINKVCGETUHYXINAJAFWQ8XTSS8
AZJZXSSPP8CKGPNZGYAZCVHHRSSWKYMAIHKULVWHLXHGFM8JDFVSZUYNXTWHOLGEZUAZWOHYLYX
CUSAFNGG UUKVWHSS8JDUCUKB8SWNGXLPZGFSTAGHKGPNZKLWYX
LIKMSHNKULOAXZYVNKULLNSZW8ZUGHAP8NAVJGFDEWYZVANKULO8ULX
EIVDGXJPTUZKMSLCFCEWNUPXTATN8KETUAUPKAXYWL8V
KXCRHPOIFGXRAGYEZBEZ8GHW8BGMLL8FXGTL NAUCFN8JDTIDVCDUJLM
CJLWZBX8AFM8SQZCGUOFNGGYKUVVJZCFAXRFQGXJVH AJULOAXZSLTOMLGET GYIGNAVJFN
LKXZLPZGFDEWYZVANKUKJDKGXZ8WBT WVLGBXHFQ8ZGHFVPFWXGYICLJFZNLJFQA DFOFSE8CLL
FXGTL NAUCFJGCAXUTUWKRTTLRZ8POFN
LWUHXGPOGXGLGYSIQZULOAYZ8Z8NZELOFVJLWSIJLWJCYMEIOD WZISPIV8XTATNS DGHS
DKUULOZULO8ULX EIVDSUFWJLXQVUSAFWM WIEWQZVLPKTVDSUFMXJQXZ8TAIBTUEYGK"

    text = "V KLF6ZXS8 KMLTWDDUAZWNZTB8ERSSIXYZW8KTUEHPNZEHPXTSWZBXVNDUTU FXGTL
NXYWYXALJZSWNGXLPZKUYVUVNGNNGJ
PZFDN8XCLOIYUTYAFYXZEMH8KWGLGAJJUVVIVOLHPOIFHHFBTY TZKZWGMKAI8NAVJYUEHGOHZGO X
```

GXRAGYEZBEZWNVJKWZISINKVCGETUHYXINAJAFWQGXTSSH  
AZJZXXSSPPFCKGPNZGYAZCVHHRSSWKYMAIHKULVWHLXHGFMMJDFVSZUYNXTWHOLGEZUAZWOHYLXY  
CUSAFNGG UUKVWHSSHJDUCUKBHSWNGXLPZGFSTAGHKGNZKLWYX  
LIKMSHNKULOAXZYVNKULLNSZWGZUGHAPGNAVJGFDEWYZVANKULOALX  
EIVDGXJPTUZKMSLCFCEWNUPXTATNKGETUAUPKAXYWLVV  
KXCRHPOIFGXRAGYEZBEZGHWGBGMLLNFXGTL NAUCFNXJDTIDVCDUJLM  
CJLWZBXZAFMGSQZCGUOFNGGYKUVVJZCFAXRFQGXJVH AJULOAXZSLTOMLGET GYIGNAVJFN  
LKXZLPZGFDEWYZVANKUKJDKGXZWZBT WVLGBXHFQGGZGHFVPFWXGYICLJFZNLJFQA DFOFSECLL  
FXGTL NAUCFJGCAXUTUWKRTTLRZSPOFN  
LWUHXGPOGXGLGYSIQZULOAYZSZZNZELOFVJLWSIJLWJCYMEIOD WZISPIVFXSTATNS DGHS  
DKUULOZULOALX EIVDSUFWJLXQVUSAFWM WIEWQZVLPKTVDSUFMXJQXZSTAIBTUEYGK"

```
cmpText = "EV KLFGZXSH KMLTWDDUAZWNZTBERSSIXYZWGKTUEHPNZEHPXTSWZBXVNDUTU  
FXGTL NXYWYXALJIZSWNGXLPZKUVYUVNGNNGJ  
PZFVDNKXCLOIYUTYAFYXZEMHXKWGLGAJJUVVIVOLHPOIFHHFBTY TZKZWGMKAIVNAVJYUEHGOHZGO X  
GXRAGYEZBEZWNVJKWZISINKVCGETUHYXINAJAFWQGXTSSH  
AZJZXXSSPPFCKGPNZGYAZCVHHRSSWKYMAIHKULVWHLXHGFMMJDFVSZUYNXTWHOLGEZUAZWOHYLXY  
CUSAFNGG UUKVWHSSHJDUCUKBHSWNGXLPZGFSTAGHKGNZKLWYX  
LIKMSHNKULOAXZYVNKULLNSZWGZUGHAPGNAVJGFDEWYZVANKULOALX  
EIVDGXJPTUZKMSLCFCEWNUPXTATNKGETUAUPKAXYWLVV  
KXCRHPOIFGXRAGYEZBEZGHWGBGMLLNFXGTL NAUCFNXJDTIDVCDUJLM  
CJLWZBXZAFMGSQZCGUOFNGGYKUVVJZCFAXRFQGXJVH AJULOAXZSLTOMLGET GYIGNAVJFN  
LKXZLPZGFDEWYZVANKUKJDKGXZWZBT WVLGBXHFQGGZGHFVPFWXGYICLJFZNLJFQA DFOFSECLL  
FXGTL NAUCFJGCAXUTUWKRTTLRZSPOFN  
LWUHXGPOGXGLGYSIQZULOAYZSZZNZELOFVJLWSIJLWJCYMEIOD WZISPIVFXSTATNS DGHS  
DKUULOZULOALX EIVDSUFWJLXQVUSAFWM WIEWQZVLPKTVDSUFMXJQXZSTAIBTUEYG"
```

length = len(text)

firstFive = ""

secondFive = ""

thirdFive = ""

forthFive = ""

fifthFive = ""

firstDict = dict()

secondDict = dict()

thirdDict = dict()

forthDict = dict()

```
fifthDict = dict()
```

```
def find_matching_chars_(self):  
    filename = 'matches.txt'  
    with open(filename, 'w') as file_object:  
        i = 0  
        while ((self.length-i) != 0):  
            count = 0  
            file_object.write("Matches# " + str(i) + ": ")  
            for j in range(self.length-i):  
                if (self.text[j] == self.cmpText[j]):  
                    if (self.text[j] == " "):  
                        file_object.write("#")  
                        count += 1  
                else:  
                    file_object.write(self.text[j])  
                    count += 1  
            file_object.write("---Match Count: " + str(count))  
            file_object.write("\n")  
            self.text = self.text[1:]  
            self.cmpText = self.cmpText[:-1:]  
            i = 1 + i
```

```
def print_Gram_lines_(self):  
    filename = 'oneGramLines.txt'  
    with open(filename, 'w') as file_object:  
        self.firstFive = self.cypher[0::5]  
        self.firstDict = Counter(self.firstFive)  
        file_object.write("Line 1: ")
```



```
for i in sorted (self.firstDict.keys()):
```

```
    l = ((i, self.firstDict[i]))
```

```
    line = str(l) + " "
```

```
    file_object.write(line)
```

```
file_object.write("\n")
```

```
print(self.firstFive + "\n")
```

```
self.secondFive = self.cypher[1::5]
```

```
self.secondDict = Counter(self.secondFive)
```

```
file_object.write("Line 2: ")
```

```
for i in sorted (self.secondDict.keys()):
```

```
    l = ((i, self.secondDict[i]))
```

```
    line = str(l) + " "
```

```
    file_object.write(line)
```

```
file_object.write("\n")
```

```
print(self.secondFive + "\n")
```

```
self.thirdFive = self.cypher[2::5]
```

```
self.thirdDict = Counter(self.thirdFive)
```

```
file_object.write("Line 3: ")
```

```
for i in sorted (self.thirdDict.keys()):
```

```
    l = ((i, self.thirdDict[i]))
```

```
    line = str(l) + " "
```

```
    file_object.write(line)
```

```
file_object.write("\n")
```

```
print(self.thirdFive + "\n")
```

```

self.forthFive = self.cypher[3::5]

self.forthDict = Counter(self.forthFive)

file_object.write("Line 4: ")

for i in sorted (self.forthDict.keys()):

    l = ((i, self.forthDict[i]))

    line = str(l) + " "

    file_object.write(line)

file_object.write("\n")


print(self.forthFive + "\n")

```

```

self.fifthFive = self.cypher[4::5]

self.fifthDict = Counter(self.fifthFive)

file_object.write("Line 5: ")

for i in sorted (self.fifthDict.keys()):

    l = ((i, self.fifthDict[i]))

    line = str(l) + " "

    file_object.write(line)

file_object.write("\n")


print(self.fifthFive + "\n")

```

```

analysis = analysis()

analysis.find_matching_chars_()

analysis.print_Gram_lines_()

```

---

```

cypher = "EV KLFZXSH KMLTWDDUAZWNZTBERSSIXYZWGKTUEHPNZEHPXOTSWZBXVNDUTU FXGTL
NXYWYXALJIZSWNGXLPZKUVYUVNGNNGJ
PZFVDNKXCLOIYUTYAFYXZEMHXKWGLGAJJUVVIVOLHPOIFHHFBTY TZKZWGMKAIVNAVJYUEHGOHZGO X
GXRAGYEZBEZWNVJKWZISINKVCGETUHYXINAJAFWQGXTSSH

```

AZJZXSSPPFCKGPNZGYAZCVHHRSSWKYMAIHKULVWHLXHGFMMJDFVSZUYNXTWHOLGEZUAZW OHYLY  
CUSAFNGG UUKVWHSSHJDUCUKBHSWNGXLPZGFSTAGHKGPNZKLWYX  
LIKMSHNKULOAXZYVNKULLNSZWGZUGHAPGNAVJGFDEWYZVANKULO AULX  
EIVDGXJPTUZKMSLCFCEWNUPXTATNKGETUAUPKAXYWL VV  
KXCRHPOIFGX RAGYEZBEZWGHWGBGMLLNFXGTL NAUCFNXJDTIDVCDUJLM  
CJLWZBXZAFMG SQZCGUOFNGGYKUVVJZCFAXRFQGXJVH AJULOAXZSLTOMLGET GYIGNAVJFN  
LKXZLPZGFDEWYZVANKUKJDKGXZWZBT WVLGBXHFQGGZGHFVPFWXGYXICLJFZNLJFQA DFOFSESCLL  
FXGTL NAUCFJGCAXUTUWKRTTLRZSPOFN  
LWUHXGPOGXGLGYSIQZULOAYZSZNZELOFVJLWSIJLWJCYMEIOD WZISPIVFX TATNS DGHS  
DKUULOZULO AULX EIVDSUFWJLXQVUSAFWM WII EWQZVLPKTVDSUFMXJQXZSTAIBTUEYGK"

numChars = 2

frequency = 0

it = 1

maxChars = 10

def get\_factors(x, temp):

for i in range(1, x + 1):

if x % i == 0:

temp.append(i)

filename = 'repeats.txt'

with open(filename, 'w') as file\_object:

while it < maxChars:

file\_object.write("\nNumber of Characters: " + str(numChars) + "\n\n")

for i in range(len(cypher)):

distance = []

for j in range(len(cypher)):

if (cypher[i:i+numChars] == cypher[j:j+numChars]):

frequency = frequency + 1

distance.append(abs(j-i))

if frequency > 1:

temp = []

```

tempDistance = 0

for t in range(len(distance)):
    if abs(distance[t]) != 0:
        tempDistance = abs(distance[t])
        break

get_factors(tempDistance,temp)

file_object.write("Chars: " + cypher[i:i+numChars] + " Frequency: " + str(frequency) + " Factors: " + str(temp) + " Distance: " + str(tempDistance) + "\n")

frequency = 0

if ((numChars + i) >= len(cypher)):
    break

numChars +=1

it +=1

```

---

```

from itertools import starmap, cycle

```

```

alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ "

```

```

letter_to_index = dict(zip(alphabet, range(len(alphabet))))

```

```

index_to_letter = dict(zip(range(len(alphabet)), alphabet))

```

```

key = "THXGV"

```

```

cypher = "EV KLFQZXS KMLTWDDUAZWNZTBERSSIXYZWGKTUEHPNZEHPXOTSWZBXVNDUTU FXGTL
NXYWYXALJIZSWNGXLPZKUVYUVNGNNGJ
PZFDNKKXCLOIYUTYAFYXZEMHXKWGLGAJJUVVIVOLHPOIFHHFBTY TZKZWGMKAIVNAVJYUEHGOHZGO X
GXRAGYEZBEZWNVJKWZISINKVCGETUHYXINAJAFWQGXTSSH
AZJZXXSSPPFCKGPNZGYAZCVHHRSSWKYMAIHKULVWHLXHGFMMDJFVSZUYNXTWHOLGEZUAZWOLYLY
CUSAFNGG UUKVWHSSHJDUCUKBHSWNGXLPZGFSTAGHKGPNZKLWYX
LIKMSHNKULOAXZYVNKULLNSZWGZUGHAPGNAVJGFDEWYZVANKULOALX
EIVDGXJPTUZKMSLCFCEWNUPXTATNKGETUAUPKAXYWLTV
KXCRHPOIFGXRAGYEZBEZGHWGBGMLLNFXGTL NAUCFNXJDTIDVCDUJLM
CJLWZBXZAFMGSSQZCGUOFNGGYKUVVJZCFAXRFQGXJVH AJULOAXZSLTOMLGET GYIGNAVJFN
LKXZLPZGFDEWYZVANKUKJDKGXWZBT WVLGBXHFQGGZGHFVFPFWXGYXICLJFZNLJFQA DFOFSESCLL
FXGTL NAUCFJGCAXUTUWKRTTLRZSPOFN
LWUHXGPOGXGLGYSIQZULOAYZSSZNZELOFVJLWSIJLWJCYMEIOD WZISPIVFXSTATNS DGHS
DKUULOZULOALX EIVDSUFWJLXQVUSAFWM WIEWQZVLPKTVDSUFMXJQXZSTAIBTUEYGK"

```

```

def split_by_(x, text):
    out = [(text[i:i+x]) for i in range(0, len(text), x)]
    print(out)

def decrypt(input, key):
    decrypted = ""
    split_input = [input[i:i + len(key)] for i in range(0, len(input), len(key))]

    for each_split in split_input:
        i = 0
        for letter in each_split:
            number = ((letter_to_index[letter] - letter_to_index[key[i]]) % len(alphabet))
            decrypted += index_to_letter[number]
            i += 1
    return decrypted

decrypted = decrypt(cypher, key)
#split_by_(5, cypher)
print(cypher)
print("\n\n")
#split_by_(5, decrypted)
print(decrypted)

```