Part 01)

1. **Submit your commented code from Part one as "Lab01_part1.py" to canvas**
    a. Attached.

2. **Explain whether you are going to conduct an exhaustive key search to determine the key or if you are going to use the English language character frequency.**
    a. Given how we already have a function that can shift our entire cipher, I find it may be easier to do an exhaustive key search using the new tool we have built.

3. **Document each potential solution you arrived at for Part 1. For each attempt of the key, you will include the N used in the shift, the corresponding alphabet, and the plaintext output from the N.**
    a. It took us eight attempts to decode the message "thisclassisverycool" for the purposes of keeping this document in some semblance of organization – I will be including the results in a separate file in PDF format.

	Alphabet:	abcdefghijklmnopqrstuvwxyz	
	Plaintext:	QEFPZIXPPFPSBOVZLLI	
	Shift by:		17
8	Ciphertext:	thisclassisverycool	

4. **Explain the difference between the two methods (exhaustive key search vs. English language character frequency) and explain how your results supported or disproved the number of attempts needed to correctly decrypt the message.**

The two methods vastly differ in their methodology – once is essentially brute forcing the ciphertext and the other is analyzing the ciphertext to establish if there are any glaring similarities to plaintext English. The exhaustive key search is exactly as it sounds, you are essentially "attempting different combinations" until you find the correct key – in this case, you are shifting-by-n until you find a plaintext solution, this method could take a maximum of 26 tries. Whereas with the English language character frequency method you analyze the most frequent letter in the ciphertext and then make educated guesses. For example, if we find that the character O is the most common in the ciphertext, we will swap it with the most common character in the English language, and any subsequent characters. Effectively, you guess one of the ciphertext character's plaintext character based on frequency and then see if the plaintext makes sense.

I would say the method I chose (exhaustive key search) and the results effectively backup the claim I made earlier in which you are potentially searching for a longer time unless you luck out. Luckily, I managed to get it on my 8th try, however it could've easily gone on for a longer time.

5. **Find a character frequency distribution for another language. Provide the frequency distribution in your lab report *not a link, but the actual distribution) and be sure to identify which language it is.**

Below is the Spanish frequency distribution. As a native speaker, this list is honestly not a surprise. If I had to guess, the frequency may be similar for other Latin-based languages.

E	13.72 %	M	3.08 %	Z	0.47 %
A	11.72 %	P	2.89 %	Á	0.44 %
O	8.44 %	B	1.49 %	É	0.36 %
S	7.20 %	H	1.18 %	Ñ	0.17 %
N	6.83 %	Q	1.11 %	X	0.14 %
R	6.41 %	Y	1.09 %	Ú	0.12 %
I	5.28 %	V	1.05 %	K	0.11 %
L	5.24 %	G	1.00 %	W	0.04 %
D	4.67 %	Ó	0.76 %	Ü	0.02 %
T	4.60 %	Í	0.70 %		
U	4.55 %	F	0.69 %		
C	3.87 %	J	0.52 %		

Part 02)

6. **Submit your commented code from Part two as "Lab01_part2.py" to canvas. Must include functions for frequency distribution and alphabet permutations. May include digram and trigram functions. If no code for digram and trigrams, hand work must be shown.**

   a. Attached.

7. **Submit your frequency analysis of each character in the ciphertext in Part2. Based on your results, what are the three most likely letters to be mapped to the plaintext character "E"?**

   Looking on, our program returns the following:

```
==============================
Here are your values:

==============================
1.) In Order:
 Counter({'I': 118, 'M': 89, 'Y': 83, 'Q': 73, 'S': 61, 'N': 61, 'F': 60, 'D': 59, 'E': 53, 'U': 52, 'J': 28, 'V': 28, 'R': 26, 'L': 18, 'T': 18, 'O': 16, 'H':
15, 'P': 14, 'G': 12, 'B': 9, 'K': 8, 'C': 6, 'A': 1, 'W': 1})
------------------------------
2.) In Frequency Format: (unordered)
M 9.8%
D 6.5%
P 1.5%
I 13.0%
L 2.0%
C 0.7%
Q 8.0%
Y 9.1%
E 5.8%
O 1.8%
F 6.6%
S 6.7%
J 3.1%
R 2.9%
N 6.7%
U 5.7%
V 3.1%
G 1.3%
K 0.9%
T 2.0%
B 1.0%
H 1.7%
A 0.1%
W 0.1%
------------------------------
Done.
PS C:\ISU\Fall22\cybe331>
```

Based on our outputs here, we find that the characters I , M, and Y are the most frequent. With this in mind, we say that the letter E is most likely one of these characters.

8. **Which method of counting digrams and trigrams did you use? Explain why you chose your method**

   I chose the sliding digram and trigram method as it was simple to implement into my already existing Python program. Additionally, I believe that with the block method, you could potentially miss a few pairings, and while the odds are low – there is the chance that the pairing you miss could be substantial.

9. Submit your digram and trigram analysis on the ciphertext. Based on your results, which combinations make reasonable substitutions?

Here is our breakdowns:

```
==============================
Here are your digram values:

==============================
1.) In Order:
 Counter({'FI': 38, 'IS': 30, 'YF': 27, 'QD': 20, 'DN': 19, 'SI': 18, 'MY': 16, 'IN': 14, 'ED': 14, 'EY': 11, 'YY': 11, 'UI': 11, 'GI': 11, 'MM': 11, 'IY': 10, 'IQ': 10, 'YU': 10, 'UU'
: 10, 'DV': 10, 'ML': 10, 'NM': 10, 'MD': 9, 'QY': 9, 'IJ': 9, 'FM': 9, 'YM': 9, 'QU': 9, 'SQ': 9, 'EN': 9, 'MN': 9, 'JQ': 8, 'UE': 8, 'QM': 8, 'DM': 8, 'QR': 7, 'RQ': 7, 'TM': 7, 'DY'
: 7, 'YQ': 7, 'IV': 6, 'JF': 6, 'MH': 6, 'NY': 6, 'FE': 6, 'FQ': 6, 'RF': 6, 'EU': 6, 'NE': 6, 'MS': 6, 'DI': 5, 'BI': 5, 'SV': 5, 'VS': 5, 'NO': 5, 'JM': 5, 'ID': 5, 'IU': 5, 'VI': 5,
 'NS': 5, 'SE': 5, 'VF': 5, 'QB': 5, 'JE': 5, 'PI': 4, 'QS': 4, 'SU': 4, 'IG': 4, 'NQ': 4, 'UN': 4, 'VQ': 4, 'PQ': 4, 'DT': 4, 'QT': 4, 'MF': 4, 'UQ': 4, 'NJ': 4, 'VM': 4, 'YE': 3, 'OI
': 3, 'MJ': 3, 'ST': 3, 'LY': 3, 'MR': 3, 'HQ': 3, 'YR': 3, 'ND': 3, 'QG': 3, 'IP': 3, 'IR': 3, 'IF': 3, 'HS': 3, 'EP': 3, 'NF': 3, 'SR': 3, 'JI': 3, 'QE': 3, 'MO': 3, 'IE': 3, 'LS': 3
, 'IH': 3, 'HM': 3, 'ME': 3, 'DE': 3, 'NN': 3, 'DP': 2, 'CM': 2, 'EO': 2, 'NI': 2, 'VE': 2, 'UJ': 2, 'RU': 2, 'UM': 2, 'NK': 2, 'KT': 2, 'MU': 2, 'SK': 2, 'RY': 2, 'TF': 2, 'RD': 2, 'L
U': 2, 'EG': 2, 'PF': 2, 'QC': 2, 'LE': 2, 'YI': 2, 'RM': 2, 'UY': 2, 'SJ': 2, 'RI': 2, 'NU': 2, 'SO': 2, 'PM': 2, 'ER': 2, 'CE': 2, 'EI': 2, 'IM': 2, 'KM': 2, 'OY': 2, 'IT': 2, 'KI':
2, 'BD': 2, 'UT': 2, 'TQ': 2, 'IC': 2, 'SM': 2, 'SD': 2, 'UH': 2, 'QV': 2, 'QJ': 2, 'IL': 1, 'LC': 1, 'DQ': 1, 'QN': 1, 'ES': 1, 'MG': 1, 'TI': 1, 'MB': 1, 'KL': 1, 'YO': 1, 'UK': 1, '
VY': 1, 'NG': 1, 'UG': 1, 'YJ': 1, 'FR': 1, 'RL': 1, 'TY': 1, 'UR': 1, 'TR': 1, 'RP': 1, 'SP': 1, 'HP': 1, 'QK': 1, 'HJ': 1, 'IO': 1, 'MT': 1, 'RE': 1, 'BQ': 1, 'TJ': 1, 'NR': 1, 'YK':
 1, 'YV': 1, 'LQ': 1, 'VJ': 1, 'UB': 1, 'NA': 1, 'AL': 1, 'YS': 1, 'SL': 1, 'LD': 1, 'HH': 1, 'HY': 1, 'CQ': 1, 'LO': 1, 'OQ': 1, 'TH': 1, 'KS': 1, 'BY': 1, 'IK': 1, 'UV': 1, 'YD': 1,
'LV': 1, 'SS': 1, 'ON': 1, 'YH': 1, 'TV': 1, 'YC': 1, 'CI': 1, 'II': 1, 'MI': 1, 'TP': 1, 'LN': 1, 'EJ': 1, 'YP': 1, 'NV': 1, 'SF': 1, 'VL': 1, 'LI': 1, 'GE': 1, 'NW': 1, 'WL': 1, 'LR'
: 1, 'HO': 1, 'YB': 1, 'JJ': 1, 'PB': 1, 'NP': 1, 'PS': 1, 'YL': 1, 'QH': 1, 'HF': 1, 'O': 1})
------------------------------
```

```
==============================
Here are your trigram values:

==============================
1.) In Order:
 Counter({'YFI': 20, 'FIS': 20, 'QDN': 19, 'MYF': 10, 'EDV': 10, 'ISI': 9, 'YYU': 9, 'YUI': 9, 'MMN': 9, 'EYY': 8, 'UEY': 7, 'SQD': 7, 'SIN': 7, 'JQR': 6, 'QMY': 6, 'DMY': 6, 'FIJ': 6,
 'NMD': 6, 'TML': 6, 'NOM': 5, 'RFI': 5, 'UIS': 5, 'INS': 5, 'NSE': 5, 'SEN': 5, 'ENE': 5, 'NED': 5, 'DVF': 5, 'VFM': 5, 'FMM': 5, 'EUU': 5, 'IJQ': 4, 'QUU': 4, 'VSQ': 4, 'DNO': 4, 'NY
F': 4, 'IJM': 4, 'MFI': 4, 'UUQ': 4, 'DNJ': 4, 'IYF': 3, 'QRQ': 3, 'SUE': 3, 'IGI': 3, 'GIS': 3, 'HQU': 3, 'SVS': 3, 'ISQ': 3, 'DNY': 3, 'YFE': 3, 'FED': 3, 'YFQ': 3, 'FQY': 3, 'YRF':
3, 'QGI': 3, 'DYM': 3, 'IFI': 3, 'ISR': 3, 'QEN': 3, 'YMF': 3, 'QBI': 3, 'IQD': 3, 'JEU': 3, 'HMS': 3, 'VIY': 3, 'FID': 3, 'IDT': 3, 'DTM': 3, 'SIQ': 3, 'EDY': 3, 'MDP': 2, 'DPI': 2, '
OIY': 2, 'SIJ': 2, 'IQS': 2, 'IVE': 2, 'JFM': 2, 'QRU': 2, 'GIN': 2, 'IST': 2, 'STM': 2, 'MDI': 2, 'BIN': 2, 'INQ': 2, 'QYF': 2, 'ISK': 2, 'TFI': 2, 'ISV': 2, 'QRD': 2, 'RDM': 2, 'QYR
': 2, 'JML': 2, 'MLU': 2, 'LUN': 2, 'UND': 2, 'NDM': 2, 'EGI': 2, 'IDY': 2, 'UNM': 2, 'FIV': 2, 'VQG': 2, 'GIF': 2, 'NFI': 2, 'UUY': 2, 'ISJ': 2, 'JIQ': 2, 'NUE': 2, 'ISO': 2, 'SOM': 2,
 'RQE': 2, 'NYM': 2, 'MNF': 2, 'SIE': 2, 'IER': 2, 'IMH': 2, 'KMY': 2, 'MYY': 2, 'YQB': 2, 'MLS': 2, 'LSV': 2, 'UQD': 2, 'IQB': 2, 'VMM': 2, 'MLY': 2, 'KIH': 2, 'IHM': 2, 'MSI': 2, 'MY
Q': 2, 'NJF': 2, 'JFI': 2, 'QSI': 2, 'VME': 2, 'MED': 2, 'UTQ': 2, 'TQD': 2, 'DNN': 2, 'NNM': 2, 'UIQ': 2, 'DVQ': 2, 'VQD': 2, 'MDY': 2, 'MSD': 2, 'IVS': 2, 'IQY': 2, 'ISU': 2, 'JMM':
2, 'QYQ': 2, 'PIL': 1, 'ILC': 1, 'LCM': 1, 'CMD': 1, 'MDQ': 1, 'DQY': 1, 'QYE': 1, 'YEO': 1, 'EOI': 1, 'RQN': 1, 'QNI': 1, 'NIQ': 1, 'QSU': 1, 'UIV': 1, 'VES': 1, 'ESU': 1, 'SUJ': 1, '
UJF': 1, 'FMJ': 1, 'MJQ': 1, 'RUM': 1, 'UMG': 1, 'MGI': 1, 'INK': 1, 'NKT': 1, 'KTI': 1, 'TIG': 1, 'TMD': 1, 'DIJ': 1, 'IJF': 1, 'FMU': 1, 'MUM': 1, 'UMM': 1, 'MMB': 1, 'MBI': 1, 'NQY
': 1, 'SKL': 1, 'KLY': 1, 'LYO': 1, 'YOM': 1, 'OMR': 1, 'MRY': 1, 'RYM': 1, 'YMH': 1, 'MHQ': 1, 'UUK': 1, 'UKT': 1, 'KTF': 1, 'DVY': 1, 'VYF': 1, 'FQG': 1, 'GIV': 1, 'VEG': 1, 'GID': 1,
 'YMY': 1, 'FIP': 1, 'IPF': 1, 'PFE': 1, 'FEU': 1, 'EUN': 1, 'PIR': 1, 'IRF': 1, 'IVQ': 1, 'SQU': 1, 'QUE': 1, 'UIP': 1, 'IPQ': 1, 'PQC': 1, 'QCM': 1, 'CMH': 1, 'MHS': 1, 'HSI': 1, 'IN
G': 1, 'NGI': 1, 'GIU': 1, 'IUG': 1, 'UGI': 1, 'GIY': 1, 'IYJ': 1, 'YJF': 1, 'JFE': 1, 'FEP': 1, 'EPF': 1, 'PFR': 1, 'FRL': 1, 'RLE': 1, 'LEY': 1, 'EYI': 1, 'YIN': 1, 'INF': 1, 'SRM':
1, 'RMJ': 1, 'MJI': 1, 'JIU': 1, 'IUU': 1, 'UYF': 1, 'NDI': 1, 'DIG': 1, 'SJI': 1, 'QSQ': 1, 'QDT': 1, 'DTY': 1, 'TYF': 1, 'DVI': 1, 'VIU': 1, 'IUR': 1, 'URI': 1, 'RIR': 1, 'IRM': 1, '
RMR': 1, 'MRF': 1, 'RQU': 1, 'QUJ': 1, 'UJQ': 1, 'JQT': 1, 'QTR': 1, 'TRP': 1, 'RPQ': 1, 'PQU': 1, 'UUI': 1, 'UIN': 1, 'INU': 1, 'MNM': 1, 'DIN': 1, 'NQT': 1, 'QTF': 1, 'SRQ': 1, 'ENY
': 1, 'ISP': 1, 'SPM': 1, 'PMO': 1, 'MOI': 1, 'OIU': 1, 'IUE': 1, 'ERQ': 1, 'RQC': 1, 'QCE': 1, 'CEI': 1, 'EIP': 1, 'IPI': 1, 'PIM': 1, 'MHP': 1, 'HPQ': 1, 'PQB': 1, 'BIQ': 1, 'DNQ': 1,
 'NQK': 1, 'QKM': 1, 'UIM': 1, 'MHJ': 1, 'HJE': 1, 'JED': 1, 'EDI': 1, 'DIY': 1, 'IYQ': 1, 'BIY': 1, 'FIO': 1, 'IOY': 1, 'OYM': 1, 'YMT': 1, 'MTM': 1, 'SRF': 1, 'FIE': 1, 'ERE': 1, 'RE
U': 1, 'NJI': 1, 'QBQ': 1, 'BQD': 1, 'FIT': 1, 'ITJ': 1, 'TJE': 1, 'UUN': 1, 'NMF': 1, 'SVM': 1, 'MNR': 1, 'NRI': 1, 'RIY': 1, 'IYM': 1, 'YML': 1, 'LYK': 1, 'YKI': 1, 'IEY': 1, 'EYV':
1, 'YVI': 1, 'IYR': 1, 'RFM': 1, 'FMY': 1, 'YQD': 1, 'MLQ': 1, 'LQS': 1, 'SIV': 1, 'IVM': 1, 'DVJ': 1, 'VJQ': 1, 'JQU': 1, 'QUB': 1, 'UBD': 1, 'BDE': 1, 'DEP': 1, 'EPI': 1, 'PIU': 1, '
IUT': 1, 'DNA': 1, 'NAL': 1, 'ALE': 1, 'LEI': 1, 'EIY': 1, 'IYU': 1, 'YUT': 1, 'MDM': 1, 'MYS': 1, 'YSL': 1, 'SLD': 1, 'LDM': 1, 'DMH': 1, 'MHH': 1, 'HHY': 1, 'HYF': 1, 'FIC': 1, 'ICQ'
: 1, 'CQY': 1, 'YFM': 1, 'FMS': 1, 'MST': 1, 'MLO': 1, 'LOQ': 1, 'OQT': 1, 'QTH': 1, 'THQ': 1, 'DNK': 1, 'NKS': 1, 'KSI': 1, 'QBY': 1, 'BYF': 1, 'FIK': 1, 'IKM': 1, 'SJE': 1, 'UUV': 1,
 'UVI': 1, 'IYD': 1, 'YDM': 1, 'MLV': 1, 'LVM': 1, 'ISS': 1, 'SSM': 1, 'SMM': 1, 'MMO': 1, 'MON': 1, 'ONM': 1, 'DYH': 1, 'YHM': 1, 'MSV': 1, 'SVI': 1, 'IYY': 1, 'YYM': 1, 'YMR': 1, 'MR
Q': 1, 'RQT': 1, 'QTV': 1, 'TVM': 1, 'MNO': 1, 'OMS': 1, 'SDE': 1, 'DED': 1, 'DYC': 1, 'YCI': 1, 'CII': 1, 'IIC': 1, 'ICE': 1, 'CED': 1, 'YMI': 1, 'MIG': 1, 'STP': 1, 'TPM': 1, 'PMS':
1, 'SDI': 1, 'DIS': 1, 'SKI': 1, 'SIT': 1, 'ITM': 1, 'MLN': 1, 'LNM': 1, 'NME': 1, 'MEY': 1, 'EYE': 1, 'YEJ': 1, 'EJE': 1, 'UYQ': 1, 'BIV': 1, 'VSI': 1, 'QYP': 1, 'YPQ': 1, 'PQS': 1, '
SIR': 1, 'IRQ': 1, 'ENU': 1, 'MNY': 1, 'DNV': 1, 'NVQ': 1, 'ISF': 1, 'SFQ': 1, 'FQD': 1, 'DNM': 1, 'MDE': 1, 'DEY': 1, 'YYF': 1, 'UEG': 1, 'INM': 1, 'NML': 1, 'LYE': 1, 'YED': 1, 'DYF
': 1, 'NFQ': 1, 'FQU': 1, 'QUH': 1, 'UHQ': 1, 'QUI': 1, 'IQV': 1, 'QVL': 1, 'VLI': 1, 'LIH': 1, 'IHS': 1, 'HSM': 1, 'SMO': 1, 'MOY': 1, 'OYF': 1, 'FIG': 1, 'IGE': 1, 'GEU': 1, 'UQV': 1,
 'QVI': 1, 'VIQ': 1, 'DNW': 1, 'NWL': 1, 'WLR': 1, 'LRY': 1, 'RYQ': 1, 'YQR': 1, 'RUE': 1, 'MNI': 1, 'NID': 1, 'DYI': 1, 'YIS': 1, 'INY': 1, 'MNQ': 1, 'NQJ': 1, 'QJM': 1, 'JMU': 1, 'MU
H': 1, 'UHO': 1, 'HOI': 1, 'MNN': 1, 'NNE': 1, 'NEN': 1, 'END': 1, 'MYB': 1, 'YBD': 1, 'BDM': 1, 'DMJ': 1, 'MJJ': 1, 'JJF': 1, 'JFQ': 1, 'YQJ': 1, 'QJE': 1, 'JEP': 1, 'EPB': 1, 'PBI':
1, 'INP': 1, 'NPS': 1, 'PSI': 1, 'QYL': 1, 'YLS': 1, 'LSI': 1, 'SIF': 1, 'RQD': 1, 'NJQ': 1, 'YQY': 1, 'YQU': 1, 'UQH': 1, 'QHS': 1, 'HSQ': 1, 'SQE': 1, 'ENM': 1, 'NMH': 1, 'MHF': 1, '
HFE': 1, 'FEO': 1, 'EO': 1, 'O': 1})
------------------------------
Done.
PS C:\ISU\Fall22\cybe331>
```

Looking at the digrams, we find that our most commonly occurring are ‘FI’, ‘IS’, and ‘YF.’ Knowing what we know about the English language – these are potentially the digram 'th,’ ‘he,’ and ‘in.’

Looking at the trigram, we have ‘YFI,’ ‘FIS,’ and ‘QDN.’ Also using what we know about the English language – we find that these could be potentially ‘the,’ ‘and,’ and ‘ing.’

Additionally, we find that there are some similarities across the single alphabetic frequency analysis, digram frequency analysis, and the trigram frequency analysis – such as ‘YF’ and ‘I” which is most likely ‘THE.’

10. **What is the plaintext message from Part 2? Show and explain how your frequency analysis led to this discovery. This includes the step-by-step process you went through to get the final answer as shown in the lab document.**
    a. As stated earlier during our frequency discussion in this lab, we found that the trigram YFI (along with the assistance of the bigram and singular character frequency analysis), is more than likely the word the. With this in mind, we swap the characters using our Python program. Here is our alphabet and ciphertext after these swaps:

```
a ==
b ==
c ==
d ==
e ==
f == h
g ==
h ==
i == e
j ==
k ==
l ==
m ==
n ==
o ==
p ==
q ==
r ==
s ==
t ==
u ==
v ==
w ==
x ==
y == t
z ==
mdpelcmdqteoethesejqrqneqsuettuevesujhmjqrumgenktegestmdejhmummbenqtheskltomrtmhquukthesvsqdnomthesqdnthesejqrdmthedvthqtrhejmlundmthqgevegedtmthepheunmdperhevq
gehesquettuepqcmhsengeugetjhephrletenhesrmjeuuthqtrhejmlundegesjeqsqdtthedveurermrhejqrqujqtrpquuenuettuesensenedvhmmnmdenqthesomthesrqentmhespmoeuettuesensened
vhmmnheseerqceepemhpqbeqdnqkmttuemhjedetqbetheotmtmlsvsqdnomthesrheereuuqdnjeqbqdnthetjeuunmhesvmmnretmltkehmseetvetrhmtqdnjhedtmlqsevmedvjqubdepeutqdnaleetutqd
nnmdmtsldmhhthecqthmstmloqthquuqdnkseqbthekmttueqdnthedtmlsvsqdnomthesjeuuvetdmthedvqdnjhedtmlvmedtmhessmmonmdthmsvettmrqtvmmnomsdedvqdnnmdtceecedtmegestpmsdesk
ehmsetmlnmetejeuutqbevseqtpqserqenuettuesensenedvhmmntmhesomthesqdnvqgeheshqdnmdetthevsqdnomthesuegenmltedthejmmnhquhqueqvlehsmothegeuuqveqdnwlrtqruettuesensene
dvhmmnedtesenthejmmnqjmuhoethesuettuesensenedvhmmnnendmtbdmjjhqtqjepbenpseqtlsehejqrqdnjqrdmtqtquuqhsqenmhheo
```

    b. Moving on, we return to our frequency analysis and find that we potentially know the two most common singular characters, leading us to the next most common character being ‘m.’ This could potentially be an ‘o’ or an ‘a.’ We swap the two respectively to see what we get with relation to the most commonly appearing letters in our ciphertext:

```
a ==
b ==
c ==
d ==
e ==
f == h
g ==
h ==
i == e
j ==
k ==
l ==
m == o
n ==
o ==
p ==
q == a
r ==
s ==
t ==
u ==
v ==
w ==
x ==
y == t
z ==
odpelcodateoethesejaraneasuettuevesujhojaruogenktegestodejhouoobenatheskltoortohauukthesvsadnoothesadnthesejardothedvthatrhejolundothagevegedtothepheunodperheva
gehesauettuepacohsengeugetjhephrletenhesrojeuuthatrhejolundegesjeasadtthedveurerorhejaraujatrpauuenuettuesensenedvhoonodenathesoothesraentohespooeuettuesensened
vhoonheseeraceepeohpabeadnakottueohjedetabetheototolsvsadnoothesrheereuuadnjeabadnthetjeuunohesvoonretoltkehoseetvetrhotadnjhedtolasevoedvjaubdepeutadnaleetutad
nnodotsldohhthecathostoloathauuadnkseabthekottueadnthedtolsvsadnoothesjeuuvetdothedvadnjhedtolvoedtohessooonodthosvettoratvoonoosdedvadnnodtceecedtoegestposdesk
ehosetolnoetejeuutabevseatpaseraenuettuesensenedvhoontohesoothesadnvageheshadnodetthevsadnoothesuegenoltedthejoonhauhaueavlehsoothegeuuaveadnwlrtaruettuesensene
dvhoonedtesenthejoonajouhoethesuettuesensenedvhoonnendotbdojjhatajepbenpseatlsehejaradnjardotatauuahsaenohheo
```

c. Continuing with our singular character analysis, we find that s and n are the next most frequently appearing characters. Combining what we see in the trigram frequencies, the letters QDN stick out. This trigram could potentially be “AND,” as we know Q is A. Here is the result of swapping these letters:

```
a ==
b ==
c ==
d == n
e ==
f == h
g ==
h ==
i == e
j ==
k ==
l ==
m == o
n == d
o ==
p ==
q == a
r ==
s ==
t ==
u ==
v ==
w ==
x ==
y == t
z ==
onpelconateoethesejaraneasuettuevesujhojaruogenktegestonejhouoobenatheskltoortohauukthesvsannoothesannthesejarnothenvthatrhejolunnothagevegentothepheunonperheva
gehesauettuepacohsengeugetjhephrletenhesrojeuuthatrhejolunnegesjeasantthenveurerorhejaraujatrpauuenuettuesensenenvhoononenathesoothesraentohespooeuettuesensenen
vhoonheseeraceepeohpabeannakottueohjenetabetheototolsvsannoothesrheereuuannjeabannthetjeuunohesvoonretoltkehoseetvetrhotannjhentolasevoenvjaubnepeutannaleetutan
nnonotslnohhthecathostoloathauuannkseabthekottueannthentolsvsannoothesjeuuvetnothenvannjhentolvoentohessooononthosvettoratvoonoosnenvannnontceecentoegestposnesk
ehosetolnoetejeuutabevseatpaseraenuettuesensenenvhoontohesoothesannvageheshannonetthevsannoothesuegenoltenthejoonhauhaueavlehsoothegeuuaveannwlrtaruettuesensene
nvhoonentesenthejoonajouhoethesuettuesensenenvhoonnennotbnojjhatajepbenpseatlsehejarannjarnotatauuahsaenohheo
```

d. In the text itself, we can begin to make a few assumptions. It appears that the word “her” is in the first line of the text – as we know “he.” This one is one of longshots, but we assume s is r for now, leaving us with the following:

```
a ==
b ==
c ==
d == n
e ==
f == h
g ==
h ==
i == e
j ==
k ==
l ==
m == o
n == d
o ==
p ==
q == a
r == c
s == r
t ==
u ==
v ==
w ==
x ==
y == t
z ==
onpelconateoetherejaranearuettueverujhojaruogenktegertonejhouoobenatherkltoortohauukthervrannootheranntherejarnothenvthatrhejolunnothagevegentothepheunonperheva
geheraeuettuepacohrengeugetjhephrletenherrojeuuthatrhejolunnegerjearantthenveurerorhejaraujatrpauuenuettuerenrenenvhoononenatherootherraentoherpooeuettuerenrenen
vhoonhereeraceepeohpabeannakottueohjenetabetheototolrvrannootherrheereuuannjeabannthetjeuunohervoonretoltkehoreetvetrhotannjhentolarevoenvjaubnepeutannaleetutan
nnonotrlnohhthecathortoloathauuannkreabthekottueannthentolrvrannootherjeuuvetnothenvannjhentolvoentoherrooononthorvettoratvoonoornenvannnontceecentoegertpornerk
ehoretolnoetejeuutabevreatpareraenuettuerenrenenvhoontoherootherannvageherhannonetthevrannootheruegenoltenthejoonhauhaueavlehroothegeuuaveannwlrtaruettuerenrene
nvhoonenterenthejoonajouhoetheruettuerenrenenvhoonnennotbnojjhatajepbenpreatlrehejarannjarnotatauuahraenohheo
```

e. We begin to see even more clarity in this ciphertext. In particular, it seems as if “Once upon a time there” is written here: onpelconateoetherejaran with this in mind, we swap the characters to (hopefully) get this right:

```
a ==
b ==
c == p
d == n
e ==
f == h
g ==
h ==
i == e
j == w
k ==
l == u
m == o
n == d
o ==
p == c
q == a
r == c
s == r
t ==
u ==
v ==
w ==
x ==
y == t
z ==
onpeuponateoetherewaranearuettueveruwhowaruogenktegertonewhouoobenatherkutoortohauukthervrannootherannttherewarnothenvthatrhewouunnothagevegentothepheunonperheva
geherauettuepapohrengeugetwhephruetenherroweuuthatrhewouunnegerwearantthenveurerorhewarauwatrpauuenuettuerenrenenvhoononenatherootherraentoherpooeuettuerenrenen
vhoonhereerapeepeohpabeannakottueohwenetabetheototourvrannootherrheereuuannweabannthetweuunohervoonretoutkehoreetvetrhotannwhentouarevoenvwaubnepeutannaueetutan
nnonotrunohhthepathortouoathauuannkreabthekottueannthentourvrannootherweuuvetnothenvannwhentouvoentoherrooononthorvettoratvoonoornenvannnontpeepentoegertpornerk
ehoretounoeteweuutabevreatpareraenuettuerenrenenvhoontoherootherannvageherhannonetthevrannootheruegenoutenthewoonhauhaueavuehroothegeuuaveannwurtaruettuerenrene
nvhoonenterenthewoonawouhoetheruettuerenrenenvhoonnennotbnowwhatawepbenpreaturehewarannwarnotatauuahraenohheo
```

f. Here, we can see we were incorrect with a few mappings, and there is some overlap with a few of our guesses - however, we can see that we're closing in on a possible solution. With this in mind, we move back to our bigrams and trigram frequencies to see if we can make any educated guesses. Here we begin to see the short comings of our python program, as we decipher further – we end up replacing some of the text we previously swapped.

```
onpelponateoethesesapaneaslettleveslshosaplovenptevest
onesholoopenathespltooptohallpthesvsannoothesanntheses
```

g. Here, we can make out "Once upon a time, there was a dear little girl" – skewed a bit due to replacing the same letter multiple times at times. Using some very old knowledge we have, this could potentially be the little red riding hood nursery.

h. To spare you some details (due to this document getting messy), I continued along with the digram analysis and found a few more parings – and was able to make out a few more words. Below is the resulting alphabet (next page).

i. The final plaintext is:

```
onceuponatimetherewasadearlittlegirlwhowaslovedbyeveryonewholookedatherbutmostofallbyhergrandmotherandtherewasnothingthatshewouldnothavegiventothechildonceshega
veheralittlecapofredvelvetwhichsuitedhersowellthatshewouldneverwearanythingelsesoshewasalwayscalledlittleredridinghoodonedayhermothersaidtohercomelittleredridin
ghoodhereisapieceofcakeandabottleofwinetakethemtoyourgrandmothersheisillandweakandtheywilldohergoodsetoutbeforeitgetshotandwhenyouaregoingwalknicelyandquietlyan
ddonotrunoffthepathoryoumayfallandbreakthebottleandthenyourgrandmotherwillgetnothingandwhenyougointoherroomdontforgettosaygoodmorninganddontpeepintoeverycornerb
eforeyoudoitiwilltakegreatcaresaidlittleredridinghoodtohermotherandgaveherhandonitthegrandmotherlivedoutinthewoodhalfaleaguefromthevillageandjustaslittleredridi
nghoodenteredthewoodawolfmetherlittleredridinghooddidnotknowwhatawickedcreaturehewasandwasnotatallafraidofhim
Would you like to replace another character? (y/n)
```

```
Here are your alphabet mappings:
a == q
b == k
c == p
d == n
e == i
f == h
g == v
h == f
i == e
j == w
k == b
l == u
m == o
n == d
o == m
p == c
q == a
r == s
s == r
t == y
u == l
v == g
w == j
x == z
y == t
z == x
```

Part 03)

11. **Submit your commented code from Part 3 as "Lab01_part3.py" to canvas. Must include encrypt and decrypt functions. Must encrypt and decrypt "The quick brown fox jumps over the lazy dog." Also, must encrypt and decrypt a message of the TA's choice.**
    Attached.