Jonathan Su

111058745

CSE-331

Task 1:

In task 1 I utilized “cycle” from itertools to be able to repeatedly sequence through each letter in the given key. From this I looped through each character in the given plaintext and properly assigned each letter in the encrypted cipher following the vigenere cipher algorithm.

Task 2:

Similar to how I solved Task 1, I again utilized “cycle” from itertools to cycle through character in the given key and repeatedly sequence through them. I then looped through each char and followed the vigenere cypher decryption algorithm.

Task 3:

Since we are given the key length in Task 3, we know exactly how many Caesar ciphers we need to break (the same amount as the key length).

Using the Chi-Squared Statistic $$\chi^2(C,E) = \sum_{i=A}^{i=Z} \frac{(C_i - E_i)^2}{E_i}$$ we can compare the frequency distribution of our subsequences to the expected English language frequency distributions. First we must divide our cipher into n strings where each string is the original string but contains every nth letter including the first letter (where n is equal to the length of the key). We then take each of those n strings and make 25 more strings where each original string was shifted 0-25(A-Z) similar to a Caesar shift decryption. We then calculate the chi-squared

statistic for each one of these 26 new strings and find the lowest value amongst them, once we have determined the lowest value we can see that the shift with the lowest value gives us our position in the alphabet (A = 0, B=1,…). After we have determined the position of the alphabet for each of the n letters in the key we can now combine those alphabet positions and spell out our key.

Task 4:

Task 4 utilizes the same underlying method as in Task 3 but with an additional beginning step. We must now first determine the key length of the given cipher and to do so I used the highest value of the Index Of Coincidences. Following this formula

$$IC = \sum_{i=A}^{i=Z} \frac{n_i(n_i - 1)}{N(N-1)}$$

we must first split the cipher into strings containing every nth(where n is the key length) character including the first character. Since we don't know the key length we start from 2 and go up, we then take each string that was produced and apply the IC formula above to each string. This gives us an IC for each string, for each n, there will be n different strings and we take the average of the ICs produced for every set of strings. After that we can compare the average IC's values with the other average IC values computed for each key length. The key lengths with the highest average IC values are the key lengths that are the most probable to be correct. In my implementation I take the highest value and second highest value key lengths and use the greatest common divisor to be my final key length.