Documentation

- *Include a PDF document outlining each of your functions, the algorithm, and why you did it that way. Is it efficient, robust etc.? How did you test it?*

Functions:

I only have three functions beside the main method.

- In the main method, it mainly does three things

  Step1: read the input txt file into "text" array and the maximum number of keys would be used - n.

  Step2: convert all low alpha of input text to Upper case and input all characters into text as well, even if it is not alpha.

  Step3: calling createSubtext methods by number of times. The calling number times is equal the number of key

- In the createSubtext method with three parameters

  @n --- the number of keys

  @len—the element numbers of input text file

  @text – the array of input text files

  Step1: according to the result of len%n, determining what the maximum length of sub cypher-text would be.

  Step2: initialize the 2D array collections of sub cypher-text into 'a', which is used for stopping checking through for-loop if there is no up case alpha exist, but index is still less than the maximum length of subtext

  Step3: input all characters of the text array into the 2D array of "collectSubtext[][]"

  Step4: create a 2D array-"freqAnaly"- 'row' represents the order of the subtext (0…n), 'col' represents the alphabet(0…25), and initialize the 2D array to zero.

Step5: loop through the 2D array- “collectSubtext”, count the frequency of all alpha in each row of “collectSubtext”. The frequency of all alpha of each row are shown on the 2D array – ‘freqAnaly’ by increasing 1 in corresponding index (row(order of key), col(A…..Z)).

Step6: Sort the freqAnaly time from higher to lower except zero. Every time the index of the current maximum frequency indicates the most frequent alpha. Then, replace all the alpha with the alpha of the CHFREQ table.

Step7: Recombine the the 2D array –‘frequenceSub’ vertically into a ‘result’ string. Print out the result string.

- indexMax(int* arr, int len) method
  @arr ------passing an array
  @len ------the length of the array
  the method is helper method to find the index of maximum value

the algorithm

Step1: create a 2D array-“freqAnaly”- ‘row’ represents the order of the subtext (0…n), ‘col’ represents the alphabet(0…25), and initialize the 2D array to zero.

Step2: loop through the 2D array- “collectSubtext”, count the frequency of all alpha in each row of “collectSubtext”. The frequency of all alpha of each row are shown on the 2D array – ‘freqAnaly’ by increasing 1 in corresponding index (row(order of key), col(A…..Z)).

Step3: Sort the freqAnaly time from higher to lower except zero. Every time the index of the current maximum frequency indicates the most frequent alpha. Then, replace all the alpha with the alpha of the CHFREQ *table.*

why you did it that way. Is it efficient, robust etc.?

I think that my way is not too bad, but I could not say that it is most efficient way. But this algorithm is easily to debug for me, since every character exists one the same corresponding index of all 2D arrays except frequency table.

How did you test it?

I used several “printf-statement” to check values of variable by passing own small special test text file.

Check one key running through the programme first to make sure that replace alphas of each subtext with correct alphas from CHFREQ table.

Check two key running through the program secondly to make sure that the process of separating text file into two subtexts correctly, meanwhile, check that the recombine process is also correct.