Written By: Jared Fowler March 2, 2015 Homework 1 Version 3 Comp 424 w/Prof. Vahab

Encrypted Message: KUHPVIBQKVOSHWHXBPOFUXHRPVLLDDWVOSKWPREDDVVIDWQRBHBGLLBBPKQUNRVOHQEIRLWOKKRDD

Decrypted Message: **BE HAPPY FOR THE MOMENT THIS MOMENT IS YOUR LIFE BY KHAYYAM OH AND ALSO THIS CLASS IS REALLY FUN**

About:

This is the third time I've made changes to my code and cracked the cipher. Please refer to my previous version submissions for more information. I've calculated this current version to be about 10 times more efficient than the previous. The difference can be seen below in the change log. Based upon a point system and a minimal "survival" score which I set, the program is able to successfully run in about 3 hours, running all keys of 10 or less, and return a single answer.

Technique Enumerated:

1. Function written to populate a hard coded dictionary. Reads dictionary and outputs in coded list format. { "zzz"}, { "zzy"}, … , {"zza"}, {"zz"}, … , {"z"} This list is purposefully stored backwards in order to do longest word comparison first. Also, there are a total of 26 hard-coded dictionaries, thuslyoptimizing the search time for a word.
2. Function written to analyze occurrences of letters in a text. Function used on complete dictionary and ciphertext. Based off the results, the function gives an intelligent guess to what the shift offset is. (Gives top 3 possibilities.)
3. Function written which handles the creation of threads and assigning the threads work. One thread is for status monitoring which prints out a percent complete bar. Other threads are used to handle the permutations and dictionary comparisons of a ciphertext. The program is written in such a way to allow the addition of more threads.
4. Each permutation thread will do the following:
   a. for every key length
      for every variable column length permutation,
         for every column order permutation
            get possible ciphertext, compare to dictionary words
               if comparison score is high enough, print to file and possibly keep copy in data section
5. By doing the above steps, utilizing both the best shift guesses and multi-threading, the returns only 1 answer, the correct one.

Version Change Log/Problems/Possible Improvements:

Version 1:

1. User word length assumptions
2. Failed to include permutations of short vs. long columns, i.e., permutation of column lengths.
3. Output all possibilities to a file, which took a significant amount of resources.
4. Basic dictionary with single word comparisons.

Version 2:

1. Dictionary written backwards in order to allow longest word comparison first.
2. Fixed and added variable column length permutation.
3. Point system included. Points += sizeof(word)^2
4. Multiple threads used.
5. Status % done bar added.
6. Changed permutation function to non-recursive method.
7. Smart guess given to simple shift and final answer.

Version 3:

1. Check was added to skip dictionary look up, or point system functions, for ciphers where the column permutation didn't have all the short columns to the right of the long columns. This speeds the program up by about 1000%. Example: 10! vs. (7!*3!)

Possible Future Improvements:

1. More variable scoring system.
2. Reverse the order of operations of individual cipher permutations and simple shifts.

References:

Almost all the submitted code was written by my own hand within the last week. The rest is as follows:
Next Permutation Algorithm: http://www.nayuki.io/page/next-lexicographical-permutation-algorithm
English Dictionary: http://www.winedt.org/Dict/
Other Reference Manuals: https://msdn.microsoft.com/en-us/library/windows/desktop/