Written By: Jared Fowler February 25, 2015 Homework 1 Version 2 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 second time I've cracked the cipher. In my previous submission I was able to successfully decipher the message by getting down to a list of 200 possibilities. After some contemplation, however, I decided to myself that I got the right answer largely due to luck. I wanted to implement a more powerful approach. Please see my previous submission for more details.

Problems in version 1:

1. User word length assumptions.
2. Failed to include the permutations of short vs. long columns which originate from incomplete columns.
3. Required output space was large.

Changes in 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.

Technique Enumerated:

1. Function written to populate a hard coded dictionary. Reads dictionary and outputs in coded list format. { "xxx"}, { "xxa"} This list is purposefully stored backwards in order to do longest word comparison first. Also, there are a total of 26 hard-coded dictionaries, thusly optimizing 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 program takes a little over an hour to run and returns only 1 answer, the correct one. (The program has been set to assume that key length is <= 8)

Possible Improvements:

1. Based upon the results of variable column length permutation, I could reduce work in the column order permutation by only allowing shorter columns to go after the permutations of full columns.
2. More variable scoring system

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/