Program 36 C program that can perform a letter frequency attack on any monoalphabetic

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define ALPHABET\_SIZE 26

// Function to calculate the letter frequency in the given text

void calculateLetterFrequency(const char \*text, int frequency[ALPHABET\_SIZE]) {

for (int i = 0; text[i] != '\0'; i++) {

char currentChar = text[i];

if (isalpha(currentChar)) {

int index = tolower(currentChar) - 'a';

frequency[index]++;

}

}

}

// Function to perform a letter frequency attack on a monoalphabetic substitution cipher

void letterFrequencyAttack(const char \*ciphertext, int topN) {

int frequency[ALPHABET\_SIZE] = {0};

// Calculate letter frequency in the ciphertext

calculateLetterFrequency(ciphertext, frequency);

// Create an array to store the frequency and corresponding letters

int frequencyArray[ALPHABET\_SIZE][2];

for (int i = 0; i < ALPHABET\_SIZE; i++) {

frequencyArray[i][0] = frequency[i];

frequencyArray[i][1] = i; // Corresponding letter index

}

// Sort the array based on frequency in descending order

for (int i = 0; i < ALPHABET\_SIZE - 1; i++) {

for (int j = i + 1; j < ALPHABET\_SIZE; j++) {

if (frequencyArray[i][0] < frequencyArray[j][0]) {

// Swap frequencies

int tempFrequency = frequencyArray[i][0];

frequencyArray[i][0] = frequencyArray[j][0];

frequencyArray[j][0] = tempFrequency;

// Swap corresponding letter indices

int tempIndex = frequencyArray[i][1];

frequencyArray[i][1] = frequencyArray[j][1];

frequencyArray[j][1] = tempIndex;

}

}

}

// Display the top N possible plaintexts

printf("Top %d Possible Plaintexts:\n", topN);

for (int i = 0; i < topN; i++) {

char suggestedChar = frequencyArray[i][1] + 'a';

printf("Suggested Plaintext #%d: Replace '%c' with '%c'\n", i + 1, suggestedChar, 'e');

}

}

int main() {

const char \*ciphertext = "Wklv lv d whvw phvvdjh.";

int topN = 5;

printf("Ciphertext: %s\n", ciphertext);

// Perform letter frequency attack and suggest possible plaintexts

letterFrequencyAttack(ciphertext, topN);

return 0;

}

Output:

