15. Write a C program that can perform a letter frequency attack on an additive cipher without human intervention. Your software should produce possible plaintexts in rough order of likelihood. It would be good if your user interface allowed the user to specify “give me the top 10 possible plaintexts.”

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX\_TEXT 1024

#define ALPHABET\_SIZE 26

const double englishFreq[ALPHABET\_SIZE] = {

8.167,

1.492,

2.782,

4.253,

12.702,

2.228,

2.015,

6.094,

6.966,

0.153,

0.772,

4.025,

2.406,

6.749,

7.507,

1.929,

0.095,

5.987,

6.327,

9.056,

2.758,

0.978,

2.360,

0.150, // X

1.974, // Y

0.074 // Z

};

typedef struct {

int key;

double score;

char plaintext[MAX\_TEXT];

} DecryptionResult;

int charToIndex(char c) {

return tolower(c) - 'a';

}

void caesarDecrypt(char\* input, char\* output, int key) {

int i = 0;

while (input[i]) {

if (isalpha(input[i])) {

char base = isupper(input[i]) ? 'A' : 'a';

output[i] = (input[i] - base - key + 26) % 26 + base;

} else {

output[i] = input[i];

}

i++;

}

output[i] = '\0';

}

double computeScore(const char\* text) {

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

int totalLetters = 0;

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

if (isalpha(text[i])) {

letterCounts[charToIndex(text[i])]++;

totalLetters++;

}

}

if (totalLetters == 0) return 0;

double score = 0;

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

double observed = (double)letterCounts[i] / totalLetters \* 100;

score += englishFreq[i] \* observed;

}

return score;

}

int compareResults(const void\* a, const void\* b) {

double diff = ((DecryptionResult\*)b)->score - ((DecryptionResult\*)a)->score;

return (diff > 0) - (diff < 0);

}

int main() {

char ciphertext[MAX\_TEXT];

int topN;

printf("Enter ciphertext (Caesar-encrypted):\n");

fgets(ciphertext, MAX\_TEXT, stdin);

ciphertext[strcspn(ciphertext, "\n")] = '\0'; // Remove newline

printf("Enter number of top possible plaintexts to display (e.g. 5 or 10): ");

scanf("%d", &topN);

DecryptionResult results[ALPHABET\_SIZE];

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

caesarDecrypt(ciphertext, results[key].plaintext, key);

results[key].key = key;

results[key].score = computeScore(results[key].plaintext);

}

qsort(results, ALPHABET\_SIZE, sizeof(DecryptionResult), compareResults);

printf("\nTop %d most likely plaintexts:\n", topN);

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

printf("\n[Key = %2d] Score = %.2f\n%s\n", results[i].key, results[i].score, results[i].plaintext);

}

return 0;

}

OUTPUT:

