16. Write a C program that can perform a letter frequency attack on any monoalphabetic substitution 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 26

#define MAX\_TRIES 26

const char englishFreqOrder[ALPHABET + 1] = "ETAOINSHRDLCUMWFGYPBVKJXQZ";

typedef struct {

char letter;

int count;

} Frequency;

typedef struct {

char plaintext[MAX\_TEXT];

double score;

} Result;

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

return ((Frequency \*)b)->count - ((Frequency \*)a)->count;

}

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

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

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

}

double computeScore(const char \*text) {

double score = 0;

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

char c = tolower(text[i]);

if (c == ' ') score += 2;

if (strchr("etaoinshrdlu", c)) score += 1;

}

return score;

}

void decryptWithKey(char \*cipher, char \*plain, char \*mapping) {

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

if (isalpha(cipher[i])) {

char c = toupper(cipher[i]);

plain[i] = isupper(cipher[i]) ? mapping[c - 'A'] : tolower(mapping[c - 'A']);

} else {

plain[i] = cipher[i];

}

}

plain[strlen(cipher)] = '\0';

}

int main() {

char ciphertext[MAX\_TEXT];

printf("Enter monoalphabetic ciphertext:\n");

fgets(ciphertext, MAX\_TEXT, stdin);

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

int topN;

printf("Enter number of top guesses to display: ");

scanf("%d", &topN);

Frequency freq[ALPHABET];

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

freq[i].letter = 'A' + i;

freq[i].count = 0;

}

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

if (isalpha(ciphertext[i])) {

freq[toupper(ciphertext[i]) - 'A'].count++;

}

}

qsort(freq, ALPHABET, sizeof(Frequency), compareFreq);

Result results[MAX\_TRIES];

for (int shift = 0; shift < MAX\_TRIES && shift < ALPHABET; shift++) {

char key[ALPHABET];

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

key[freq[i].letter - 'A'] = englishFreqOrder[(i + shift) % ALPHABET];

}

decryptWithKey(ciphertext, results[shift].plaintext, key);

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

}

qsort(results, MAX\_TRIES, sizeof(Result), compareResult);

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

for (int i = 0; i < topN && i < MAX\_TRIES; i++) {

printf("\n[%d] Score: %.2f\n%s\n", i + 1, results[i].score, results[i].plaintext);

}

return 0;

}

OUTPUT:

