ISS Assignment 2

## Name- Manish kumar

## Enrollment- 2020ITB007

# Shift Cipher Encryption

#include<stdio.h>

*int* main(){

printf("Welcome! This is a Shift Cipher!!!\n");

*int* k;

printf("Enter the key: ");

scanf("%d",&k);

*char* str[256];

*char* cipher[256];

printf("Enter the string: ");

scanf("%s",str);

for(*int* i=0;i<256;i++){

if((str[i]>=65 && str[i] <= 90)){

*int* temp = str[i] - 65;

temp = (k+temp)%26;

cipher[i] = temp + 65;

}

if((str[i]>=97 && str[i] <= 122)){

*int* temp = str[i] - 97;

temp = (k+temp)%26;

cipher[i] = temp + 97;

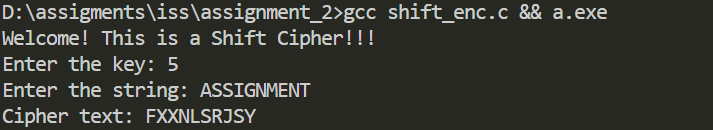
}

}

printf("Cipher text: %s\n",cipher);

return 0;

}



# Shift Cipher Decryption

#include<stdio.h>

#include<string.h>

*int* addInverse(*int* *x*, *int* *m*) {

*int* addInv = (m - (x % m));

if(addInv==26)

return 0;

return addInv;

}

*void* decrypt(*char* *s*[], *int* *key*){

*int* key\_inverse = addInverse(key, 26);

for(*int* i=0; i<strlen(s); i++){

if(s[i]>=65 && s[i]<=90)

s[i] = (*char*)(((*int*)s[i]-65 + key\_inverse)%26 + 65);

else

s[i] = (*char*)(((*int*)s[i]-97 + key\_inverse)%26 + 97);

}

}

*int* main(){

*int* n;

printf("Enter the size of cipher-text: ");

scanf("%d", &n);

*char* s[n];

printf("\nEnter the cipher-text to decrypt: ");

scanf("%s", s);

*int* key;

printf("\nEnter the value of key: ");

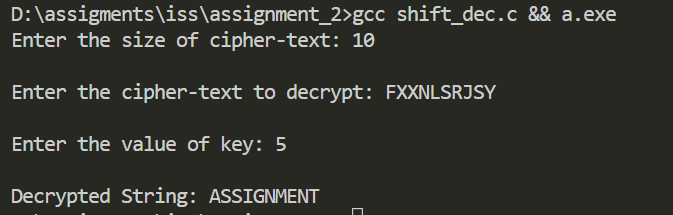
scanf("%d", &key);

decrypt(s, key);

printf("\nDecrypted String: %s", s);

return 0;

}



# Multiplicative Cipher Encryption

#include<stdio.h>

#include<string.h>

*void* encrypt(*char* *s*[], *int* *key*) {

for (*int* i = 0; i < strlen(s); i++) {

if (s[i] >= 65 && s[i] <= 90)

s[i] = (*char*)((key \* (*int*)(s[i] - 65)) % 26 + 65);

else

s[i] = (*char*)((key \* (*int*)(s[i] - 97)) % 26 + 97);

}

}

*int* main() {

*int* n;

printf("Enter the size of string to encrypt: ");

scanf("%d", & n);

*char* s[n];

printf("\nEnter the string to Encrypt: ");

scanf("%s", s);

*int* key;

printf("\nEnter the value of key: ");

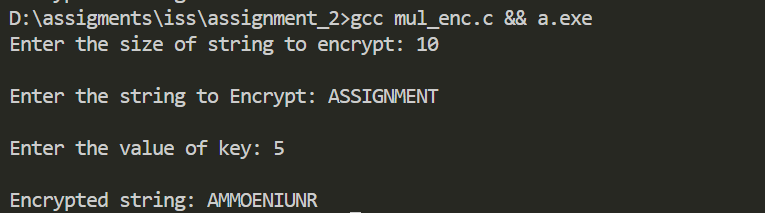
scanf("%d", & key);

encrypt(s, key);

printf("\nEncrypted string: %s", s);

}

Output



# Multiplicative Cipher Decryption

#include<stdio.h>

#include<string.h>

*int* mulInverse(*int* *k*) {

for (*int* i = 0; i < 26; i++) {

*int* flag = (i \* k) % 26;

if (flag == 1) {

return i;

}

}

return -1;

}

*void* decrypt(*char* *s*[], *int* *key*) {

*int* key\_inverse = mulInverse(key);

if (key\_inverse == -1) {

printf("Invalid Key!");

return;

}

printf("Key Inverse: %d\n", key\_inverse);

for (*int* i = 0; i < strlen(s); i++) {

if (s[i] >= 65 && s[i] <= 90)

s[i] = (*char*)((key\_inverse \* (*int*) s[i] - 65) % 26 + 65);

else

s[i] = (*char*)((key\_inverse \* (*int*) s[i] - 97) % 26 + 97);

}

}

*int* main() {

*int* n;

printf("Enter the size of cipher-text: ");

scanf("%d", & n);

*char* s[n];

printf("\nEnter the cipher-text to decrypt: ");

scanf("%s", s);

*int* key;

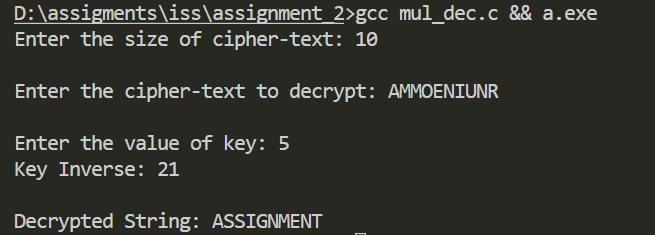
printf("\nEnter the value of key: ");

scanf("%d", & key);

decrypt(s, key);

printf("\nDecrypted String: %s", s);

}



# Affine Cipher Encryption

#include<stdio.h>

#include<string.h>

*int* gcd(*int* *a*, *int* *m*) {

while (a != m) {

if (a > m)

a -= m;

else

m -= a;

}

return a;

}

*void* encrypt(*char* *s*[], *int* *addkey*, *int* *mulkey*) {

for (*int* i = 0; i < strlen(s); i++) {

if (s[i] >= 65 && s[i] <= 90)

s[i] = (*char*)(((mulkey \* (*int*)(s[i] - 65)) + addkey) % 26 + 65);

else

s[i] = (*char*)(((mulkey \* (*int*)(s[i] - 97)) + addkey) % 26 + 97);

}

}

*int* main() {

*int* n;

printf("Enter the size of string to encrypt: ");

scanf("%d", & n);

*char* s[n];

printf("\nEnter the string to Encrypt: ");

scanf("%s", s);

*int* addkey;

printf("\nEnter the value of key to add: ");

scanf("%d", & addkey);

*int* mulkey;

printf("\nEnter the value of key to multiply: ");

scanf("%d", & mulkey);

if (gcd(mulkey, 26) != 1) {

printf("Invalid Key! Modular multiplicative inverse doesn't exist.");

return 0;

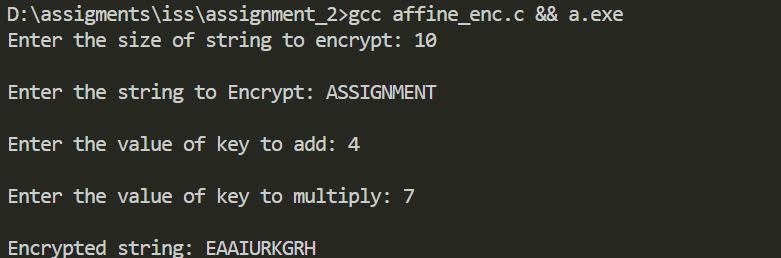
}

encrypt(s, addkey, mulkey);

printf("\nEncrypted string: %s", s);

return 0;

}



# Affine Cipher Decryption

#include<stdio.h>

#include<string.h>

*int* addInverse(*int* *x*, *int* *m*) {

*int* addInv = (m - (x % m));

if (addInv == 26)

return 0;

return addInv;

}

*int* mulInverse(*int* *k*) {

for (*int* i = 0; i < 26; i++) {

*int* flag = (i \* k) % 26;

if (flag == 1) {

return i;

}

}

return -1;

}

*void* decrypt(*char* *s*[], *int* *addkey*, *int* *mulkey*) {

*int* addkey\_inverse = addInverse(addkey, 26);

*int* mulkey\_inverse = mulInverse(mulkey);

for (*int* i = 0; i < strlen(s); i++) {

if (s[i] >= 65 && s[i] <= 90)

s[i] = (*char*)((mulkey\_inverse \* ((*int*)(s[i] - 65) +

addkey\_inverse)) % 26 + 65);

else

s[i] = (*char*)((mulkey\_inverse \* ((*int*)(s[i] - 97) +

addkey\_inverse)) % 26 + 97);

}

}

*int* main() {

*int* n;

printf("Enter the size of cipher-text: ");

scanf("%d", & n);

*char* s[n];

printf("\nEnter the cipher-text to decrypt: ");

scanf("%s", s);

*int* addkey;

printf("\nEnter the value of key to add: ");

scanf("%d", & addkey);

*int* mulkey;

printf("\nEnter the value of key to multiply: ");

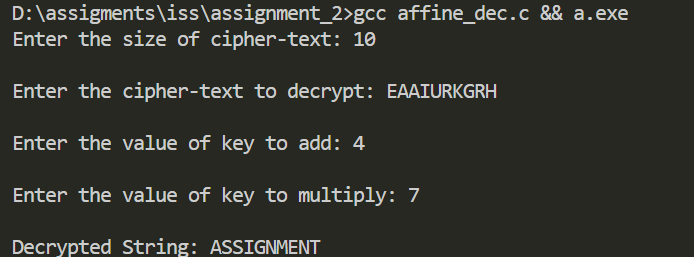
scanf("%d", & mulkey);

decrypt(s, addkey, mulkey);

printf("\nDecrypted String: %s", s);

return 0;

}



# Playfair Cipher Encryption

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

*void* toLowerCase(*char* *s*[], *int* *ps*) {

for (*int* i = 0; i < ps; i++) {

if (s[i] >= 65 && s[i] <= 90)

s[i] += 32;

}

}

*void* keyTable(*char* *key*[], *int* *ks*, *char* keyT[5][5]) {

*int* i, j, k, flag = 0;

*int* \* map;

map = (*int* \* ) calloc(26, sizeof(*int*));

for (i = 0; i < ks; i++) {

if (key[i] != 'j')

map[key[i] - 97] = 2;

}

map['j' - 97] = 1;

i = 0;

j = 0;

for (k = 0; k < ks; k++) {

if (map[key[k] - 97] == 2) {

map[key[k] - 97] -= 1;

keyT[i][j] = key[k];

j++;

if (j == 5) {

i++;

j = 0;

}

}

}

for (k = 0; k < 26; k++) {

if (map[k] == 0) {

keyT[i][j] = (*char*)(k + 97);

j++;

if (j == 5) {

i++;

j = 0;

}

}

}

}

*void* search(*char* keyT[5][5], *char* *a*, *char* *b*, *int* *arr*[]) {

*int* i, j;

if (a == 'j')

a = 'i';

else if (b == 'j')

b = 'i';

for (i = 0; i < 5; i++) {

for (j = 0; j < 5; j++) {

if (keyT[i][j] == a) {

arr[0] = i;

arr[1] = j;

} else if (keyT[i][j] == b) {

arr[2] = i;

arr[3] = j;

}

}

}

}

*void* encrypt(*char* *s*[], *char* *key*[]) {

*char* keyT[5][5];

*int* ks = strlen(key);

toLowerCase(key, ks);

*int* ps = strlen(s);

toLowerCase(s, ps);

if (ps % 2 != 0) {

s[ps++] = 'z';

s[ps] = '\0';

}

keyTable(key, ks, keyT);

*int* i, a[4];

for (i = 0; i < ps; i += 2) {

search(keyT, s[i], s[i + 1], a);

if (a[0] == a[2]) {

s[i] = keyT[a[0]][(a[1] + 1) % 5];

s[i + 1] = keyT[a[0]][(a[3] + 1) % 5];

} else if (a[1] == a[3]) {

s[i] = keyT[(a[0] + 1) % 5][a[1]];

s[i + 1] = keyT[(a[2] + 1) % 5][a[1]];

} else {

s[i] = keyT[a[0]][a[3]];

s[i + 1] = keyT[a[2]][a[1]];

}

}

}

*int* main() {

*int* n;

printf("Enter the size of string to encrypt: ");

scanf("%d", & n);

*char* s[n];

printf("\nEnter the string to Encrypt: ");

scanf("%s", s);

*int* keySize;

printf("\nEnter the key size: ");

scanf("%d", & keySize);

*char* key[keySize];

printf("\nEnter the key text: ");

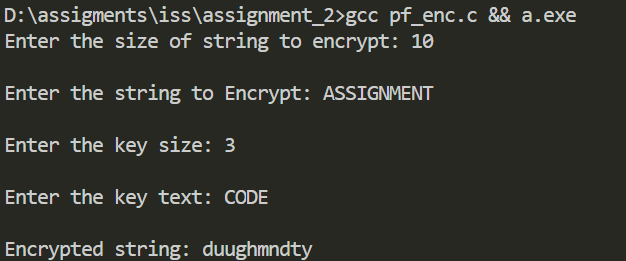
scanf("%s", key);

encrypt(s, key);

printf("\nEncrypted string: %s", s);

return 0;

}



# Playfair Cipher Decryption

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

*void* toLowerCase(*char* *s*[], *int* *ps*) {

for (*int* i = 0; i < ps; i++) {

if (s[i] >= 65 && s[i] <= 90)

s[i] += 32;

}

}

*void* keyTable(*char* *key*[], *int* *ks*, *char* keyT[5][5]) {

*int* i, j, k, flag = 0;

*int* \* map;

map = (*int* \* ) calloc(26, sizeof(*int*));

for (i = 0; i < ks; i++) {

if (key[i] != 'j')

map[key[i] - 97] = 2;

}

map['j' - 97] = 1;

i = 0;

j = 0;

for (k = 0; k < ks; k++) {

if (map[key[k] - 97] == 2) {

map[key[k] - 97] -= 1;

keyT[i][j] = key[k];

j++;

if (j == 5) {

i++;

j = 0;

}

}

}

for (k = 0; k < 26; k++) {

if (map[k] == 0) {

keyT[i][j] = (*char*)(k + 97);

j++;

if (j == 5) {

i++;

j = 0;

}

}

}

}

*int* mod5(*int* *a*) {

if (a < 0)

a += 5;

return (a % 5);

}

*void* search(*char* keyT[5][5], *char* *a*, *char* *b*, *int* *arr*[]) {

*int* i, j;

if (a == 'j')

a = 'i';

else if (b == 'j')

b = 'i';

for (i = 0; i < 5; i++) {

for (j = 0; j < 5; j++) {

if (keyT[i][j] == a) {

arr[0] = i;

arr[1] = j;

} else if (keyT[i][j] == b) {

arr[2] = i;

arr[3] = j;

}

}

}

}

*void* decrypt(*char* *s*[], *char* *key*[]) {

*char* keyT[5][5];

*int* ks = strlen(key);

toLowerCase(key, ks);

*int* ps = strlen(s);

toLowerCase(s, ps);

keyTable(key, ks, keyT);

*int* i, a[4];

for (i = 0; i < ps; i += 2) {

search(keyT, s[i], s[i + 1], a);

if (a[0] == a[2]) {

s[i] = keyT[a[0]][mod5(a[1] - 1)];

s[i + 1] = keyT[a[0]][mod5(a[3] - 1)];

} else if (a[1] == a[3]) {

s[i] = keyT[mod5(a[0] - 1)][a[1]];

s[i + 1] = keyT[mod5(a[2] - 1)][a[1]];

} else {

s[i] = keyT[a[0]][a[3]];

s[i + 1] = keyT[a[2]][a[1]];

}

}

}

*int* main() {

*int* n;

printf("Enter the size of cipher-text: ");

scanf("%d", & n);

*char* s[n];

printf("\nEnter the cipher-text to decrypt: ");

scanf("%s", s);

*int* keySize;

printf("\nEnter the key size: ");

scanf("%d", & keySize);

*char* key[keySize];

printf("\nEnter the key text: ");

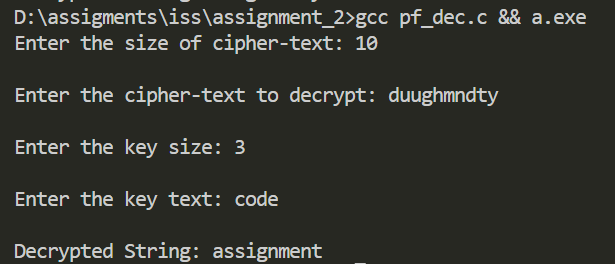
scanf("%s", key);

decrypt(s, key);

printf("\nDecrypted String: %s", s);

return 0;

}



# Hill Cipher Encryption

#include <stdio.h>

#include <string.h>

// Function to encrypt the plaintext using the Hill cipher

*void* hillCipherEncrypt(*char* *plaintext*[], *int* keyMatrix[2][2]) {

*int* len = strlen(plaintext);

*int* encrypted[len];

for (*int* i = 0; i < len; i += 2) {

*int* x = plaintext[i] - 'A';

*int* y = plaintext[i + 1] - 'A';

encrypted[i] = (keyMatrix[0][0] \* x + keyMatrix[0][1] \* y) % 26;

encrypted[i + 1] = (keyMatrix[1][0] \* x + keyMatrix[1][1] \* y) % 26;

}

printf("Encrypted text: ");

for (*int* i = 0; i < len; i++) {

printf("%c", encrypted[i] + 'A');

}

printf("\n");

}

*int* main() {

*char* plaintext[100];

*int* keyMatrix[2][2];

printf("Enter the plaintext (in uppercase, no spaces): ");

scanf("%s", plaintext);

printf("Enter the 2x2 key matrix (e.g., '3 2 5 7' for '3 2; 5 7'): ");

for (*int* i = 0; i < 2; i++) {

for (*int* j = 0; j < 2; j++) {

scanf("%d", &keyMatrix[i][j]);

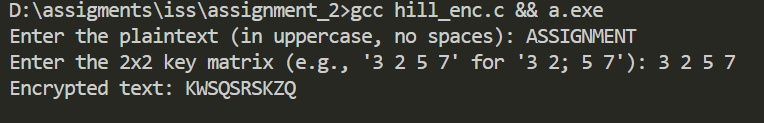
}

}

hillCipherEncrypt(plaintext, keyMatrix);

return 0;

}



# Hill Cipher Decryption

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <math.h>

*int* modInverse(*int* *a*, *int* *m*) {

a = a % m;

for (*int* x = 1; x < m; x++) {

if ((a \* x) % m == 1) {

return x;

}

}

return -1; // No modular inverse

}

*void* hillCipherDecrypt(*char* *ciphertext*[], *int* keyMatrix[2][2]) {

*int* len = strlen(ciphertext);

*int* decrypted[len];

// Calculate the determinant of the key matrix

*int* det = keyMatrix[0][0] \* keyMatrix[1][1] - keyMatrix[0][1] \* keyMatrix[1][0];

// Calculate the modular inverse of the determinant

*int* modInvDet = modInverse(det, 26);

if (modInvDet == -1) {

printf("Cannot find the modular inverse of the determinant. Decryption not possible.\n");

return;

}

// Calculate the adjugate matrix

*int* adjugate[2][2];

adjugate[0][0] = keyMatrix[1][1];

adjugate[0][1] = -keyMatrix[0][1];

adjugate[1][0] = -keyMatrix[1][0];

adjugate[1][1] = keyMatrix[0][0];

// Calculate the inverse key matrix

for (*int* i = 0; i < 2; i++) {

for (*int* j = 0; j < 2; j++) {

adjugate[i][j] = (adjugate[i][j] + 26) % 26;

adjugate[i][j] = (adjugate[i][j] \* modInvDet) % 26;

}

}

// Decrypt the ciphertext

for (*int* i = 0; i < len; i += 2) {

*int* x = ciphertext[i] - 'A';

*int* y = ciphertext[i + 1] - 'A';

decrypted[i] = (adjugate[0][0] \* x + adjugate[0][1] \* y) % 26;

decrypted[i + 1] = (adjugate[1][0] \* x + adjugate[1][1] \* y) % 26;

}

printf("Decrypted text: ");

for (*int* i = 0; i < len; i++) {

printf("%c", decrypted[i] + 'A');

}

printf("\n");

}

*int* main() {

*char* ciphertext[100];

*int* keyMatrix[2][2];

printf("Enter the ciphertext (in uppercase, no spaces): ");

scanf("%s", ciphertext);

printf("Enter the 2x2 key matrix (e.g., '3 2 5 7' for '3 2; 5 7'): ");

for (*int* i = 0; i < 2; i++) {

for (*int* j = 0; j < 2; j++) {

scanf("%d", &keyMatrix[i][j]);

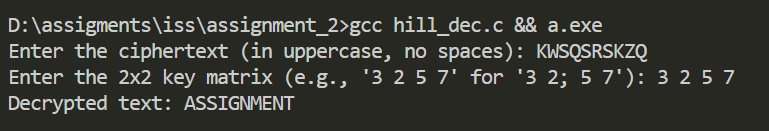
}

}

hillCipherDecrypt(ciphertext, keyMatrix);

return 0;

}



## Hence use an exhaustive key search to decrypt the following ciphertext, which was

## encrypted using a Shift Cipher:

## BMMTDXLTANZXXYYHKMMHYKXXRHNKLXEYYKHFFXFHKR

#include <stdio.h>

#include <string.h>

*void* hack(*char* *ciphertext*[], *int* *key*) {

*int* length = strlen(ciphertext);

for (*int* i = 0; i < length; i++) {

*char* c = ciphertext[i];

if (c >= 'A' && c <= 'Z') {

c = ((c - 'A' - key + 26) % 26) + 'A';

}

putchar(c);

}

printf("\n");

}

*int* main() {

*char* ciphertext[] =

"BMMTDXLTANZXXYYHKMMHYKXXRHNKLXEYYKHFFXFHKR";

printf("Shift Cipher Decryption:\n");

for (*int* key = 0; key <= 25; key++) {

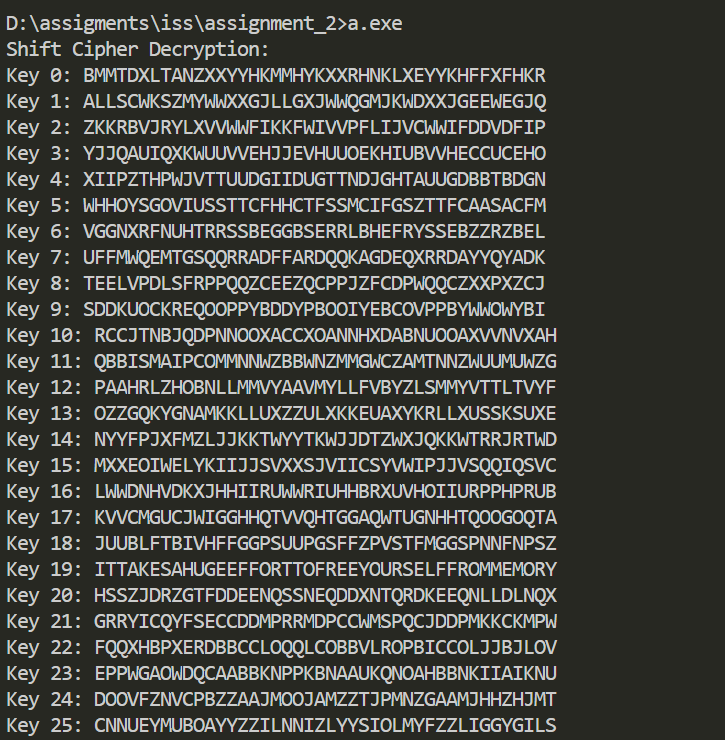
printf("Key %d: ", key);

hack(ciphertext, key);

}

return 0;

}



## Use an exhaustive key search to decrypt the following ciphertext, which was encrypted

## using a Multiplicative Cipher:

## WFEJBYOFAJZEYDCMRBKJRKWABKXSWKJZSFQ

#include <stdio.h>

#include <string.h>

*int* mulInverse(*int* *k*) {

for (*int* i = 0; i < 26; i++) {

*int* flag = (i \* k) % 26;

if (flag == 1) {

return i;

}

}

return -1;

}

*void* hack(*char* *ciphertext*[], *int* *key*) {

*int* length = strlen(ciphertext);

for (*int* i = 0; i < length; i++) {

*char* c = ciphertext[i];

if (c >= 'A' && c <= 'Z') {

c = ((c - 'A') \* key) % 26 + 'A';

}

putchar(c);

}

printf("\n");

}

*int* main() {

*char* ciphertext[] = "WFEJBYOFAJZEYDCMRBKJRKWABKXSWKJZSFQ";

printf("Multiplicative Cipher Decryption:\n");

for (*int* key = 1; key <= 26; key++) {

if (mulInverse(key) == -1)

continue;

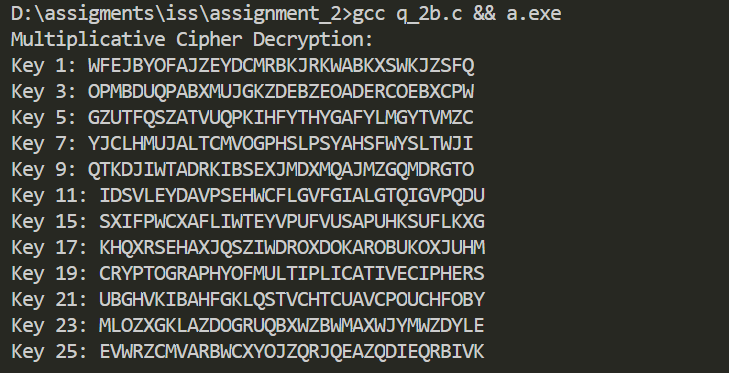
printf("Key %d: ", key);

hack(ciphertext, key);

}

return 0;

}



## Use an exhaustive key search to decrypt the following ciphertext, which was encrypted

## using a Affine Cipher:

## EFXECFBDQGGXRADQTFFUFSPGAHQTDGGAFZDJFGHJFBDQGHGDCCGXSFJDHQGAFZDJF

#include <stdio.h>

#include <string.h>

*int* modInverse(*int* *a*, *int* *m*) {

for (*int* x = 1; x < m; x++) {

if ((a \* x) % m == 1) {

return x;

}

}

return -1;

}

*void* hack(*char* *ciphertext*[], *int* *key1*, *int* *key2*) {

*int* length = strlen(ciphertext);

*int* modInv = modInverse(key1, 26);

if (modInv == -1) {

return;

}

printf("Key 1=%d, Key 2=%d: ", key1, key2);

for (*int* i = 0; i < length; i++) {

*char* c = ciphertext[i];

if (c >= 'A' && c <= 'Z') {

c = ((modInv \* (c - 'A' - key2 + 26)) % 26) + 'A';

}

putchar(c);

}

printf("\n");

}

*int* main() {

*char* ciphertext[] =

"EFXECFBDQGGXRADQTFFUFSPGAHQTDGGAFZDJFGHJFBDQGHGDCCGXSFJDHQGAFZDJF";

printf("Affine Cipher Decryption:\n");

for (*int* key1 = 1; key1 <= 25; key1++) {

for (*int* key2 = 0; key2 < 26; key2++) {

hack(ciphertext, key1, key2);

}

}

return 0;

}

Affine Cipher Decryption:

Key 1=1, Key 2=0: EFXECFBDQGGXRADQTFFUFSPGAHQTDGGAFZDJFGHJFBDQGHGDCCGXSFJDHQGAFZDJF

Key 1=1, Key 2=1: DEWDBEACPFFWQZCPSEETEROFZGPSCFFZEYCIEFGIEACPFGFCBBFWREICGPFZEYCIE

Key 1=1, Key 2=2: CDVCADZBOEEVPYBORDDSDQNEYFORBEEYDXBHDEFHDZBOEFEBAAEVQDHBFOEYDXBHD

Key 1=1, Key 2=3: BCUBZCYANDDUOXANQCCRCPMDXENQADDXCWAGCDEGCYANDEDAZZDUPCGAENDXCWAGC

Key 1=1, Key 2=4: ABTAYBXZMCCTNWZMPBBQBOLCWDMPZCCWBVZFBCDFBXZMCDCZYYCTOBFZDMCWBVZFB

Key 1=1, Key 2=5: ZASZXAWYLBBSMVYLOAAPANKBVCLOYBBVAUYEABCEAWYLBCBYXXBSNAEYCLBVAUYEA

Key 1=1, Key 2=6: YZRYWZVXKAARLUXKNZZOZMJAUBKNXAAUZTXDZABDZVXKABAXWWARMZDXBKAUZTXDZ

Key 1=1, Key 2=7: XYQXVYUWJZZQKTWJMYYNYLIZTAJMWZZTYSWCYZACYUWJZAZWVVZQLYCWAJZTYSWCY

Key 1=1, Key 2=8: WXPWUXTVIYYPJSVILXXMXKHYSZILVYYSXRVBXYZBXTVIYZYVUUYPKXBVZIYSXRVBX

Key 1=1, Key 2=9: VWOVTWSUHXXOIRUHKWWLWJGXRYHKUXXRWQUAWXYAWSUHXYXUTTXOJWAUYHXRWQUAW

Key 1=1, Key 2=10: UVNUSVRTGWWNHQTGJVVKVIFWQXGJTWWQVPTZVWXZVRTGWXWTSSWNIVZTXGWQVPTZV

Key 1=1, Key 2=11: TUMTRUQSFVVMGPSFIUUJUHEVPWFISVVPUOSYUVWYUQSFVWVSRRVMHUYSWFVPUOSYU

Key 1=1, Key 2=12: STLSQTPREUULFOREHTTITGDUOVEHRUUOTNRXTUVXTPREUVURQQULGTXRVEUOTNRXT

Key 1=1, Key 2=13: RSKRPSOQDTTKENQDGSSHSFCTNUDGQTTNSMQWSTUWSOQDTUTQPPTKFSWQUDTNSMQWS

Key 1=1, Key 2=14: QRJQORNPCSSJDMPCFRRGREBSMTCFPSSMRLPVRSTVRNPCSTSPOOSJERVPTCSMRLPVR

Key 1=1, Key 2=15: PQIPNQMOBRRICLOBEQQFQDARLSBEORRLQKOUQRSUQMOBRSRONNRIDQUOSBRLQKOUQ

Key 1=1, Key 2=16: OPHOMPLNAQQHBKNADPPEPCZQKRADNQQKPJNTPQRTPLNAQRQNMMQHCPTNRAQKPJNTP

Key 1=1, Key 2=17: NOGNLOKMZPPGAJMZCOODOBYPJQZCMPPJOIMSOPQSOKMZPQPMLLPGBOSMQZPJOIMSO

Key 1=1, Key 2=18: MNFMKNJLYOOFZILYBNNCNAXOIPYBLOOINHLRNOPRNJLYOPOLKKOFANRLPYOINHLRN

Key 1=1, Key 2=19: LMELJMIKXNNEYHKXAMMBMZWNHOXAKNNHMGKQMNOQMIKXNONKJJNEZMQKOXNHMGKQM

Key 1=1, Key 2=20: KLDKILHJWMMDXGJWZLLALYVMGNWZJMMGLFJPLMNPLHJWMNMJIIMDYLPJNWMGLFJPL

Key 1=1, Key 2=21: JKCJHKGIVLLCWFIVYKKZKXULFMVYILLFKEIOKLMOKGIVLMLIHHLCXKOIMVLFKEIOK

Key 1=1, Key 2=22: IJBIGJFHUKKBVEHUXJJYJWTKELUXHKKEJDHNJKLNJFHUKLKHGGKBWJNHLUKEJDHNJ

Key 1=1, Key 2=23: HIAHFIEGTJJAUDGTWIIXIVSJDKTWGJJDICGMIJKMIEGTJKJGFFJAVIMGKTJDICGMI

Key 1=1, Key 2=24: GHZGEHDFSIIZTCFSVHHWHURICJSVFIICHBFLHIJLHDFSIJIFEEIZUHLFJSICHBFLH

Key 1=1, Key 2=25: FGYFDGCERHHYSBERUGGVGTQHBIRUEHHBGAEKGHIKGCERHIHEDDHYTGKEIRHBGAEKG

Key 1=3, Key 2=0: KTZKSTJBOCCZXABOPTTYTGFCALOPBCCATRBDTCLDTJBOCLCBSSCZGTDBLOCATRBDT

Key 1=3, Key 2=1: BKQBJKASFTTQORSFGKKPKXWTRCFGSTTRKISUKTCUKASFTCTSJJTQXKUSCFTRKISUK

Key 1=3, Key 2=2: SBHSABRJWKKHFIJWXBBGBONKITWXJKKIBZJLBKTLBRJWKTKJAAKHOBLJTWKIBZJLB

Key 1=3, Key 2=3: JSYJRSIANBBYWZANOSSXSFEBZKNOABBZSQACSBKCSIANBKBARRBYFSCAKNBZSQACS

Key 1=3, Key 2=4: AJPAIJZRESSPNQREFJJOJWVSQBEFRSSQJHRTJSBTJZRESBSRIISPWJTRBESQJHRTJ

Key 1=3, Key 2=5: RAGRZAQIVJJGEHIVWAAFANMJHSVWIJJHAYIKAJSKAQIVJSJIZZJGNAKISVJHAYIKA

Key 1=3, Key 2=6: IRXIQRHZMAAXVYZMNRRWREDAYJMNZAAYRPZBRAJBRHZMAJAZQQAXERBZJMAYRPZBR

Key 1=3, Key 2=7: ZIOZHIYQDRROMPQDEIINIVURPADEQRRPIGQSIRASIYQDRARQHHROVISQADRPIGQSI

Key 1=3, Key 2=8: QZFQYZPHUIIFDGHUVZZEZMLIGRUVHIIGZXHJZIRJZPHUIRIHYYIFMZJHRUIGZXHJZ

Key 1=3, Key 2=9: HQWHPQGYLZZWUXYLMQQVQDCZXILMYZZXQOYAQZIAQGYLZIZYPPZWDQAYILZXQOYAQ

Key 1=3, Key 2=10: YHNYGHXPCQQNLOPCDHHMHUTQOZCDPQQOHFPRHQZRHXPCQZQPGGQNUHRPZCQOHFPRH

Key 1=3, Key 2=11: PYEPXYOGTHHECFGTUYYDYLKHFQTUGHHFYWGIYHQIYOGTHQHGXXHELYIGQTHFYWGIY

Key 1=3, Key 2=12: GPVGOPFXKYYVTWXKLPPUPCBYWHKLXYYWPNXZPYHZPFXKYHYXOOYVCPZXHKYWPNXZP

Key 1=3, Key 2=13: XGMXFGWOBPPMKNOBCGGLGTSPNYBCOPPNGEOQGPYQGWOBPYPOFFPMTGQOYBPNGEOQG

Key 1=3, Key 2=14: OXDOWXNFSGGDBEFSTXXCXKJGEPSTFGGEXVFHXGPHXNFSGPGFWWGDKXHFPSGEXVFHX

Key 1=3, Key 2=15: FOUFNOEWJXXUSVWJKOOTOBAXVGJKWXXVOMWYOXGYOEWJXGXWNNXUBOYWGJXVOMWYO

Key 1=3, Key 2=16: WFLWEFVNAOOLJMNABFFKFSROMXABNOOMFDNPFOXPFVNAOXONEEOLSFPNXAOMFDNPF

Key 1=3, Key 2=17: NWCNVWMERFFCADERSWWBWJIFDORSEFFDWUEGWFOGWMERFOFEVVFCJWGEORFDWUEGW

Key 1=3, Key 2=18: ENTEMNDVIWWTRUVIJNNSNAZWUFIJVWWUNLVXNWFXNDVIWFWVMMWTANXVFIWUNLVXN

Key 1=3, Key 2=19: VEKVDEUMZNNKILMZAEEJERQNLWZAMNNLECMOENWOEUMZNWNMDDNKREOMWZNLECMOE

Key 1=3, Key 2=20: MVBMUVLDQEEBZCDQRVVAVIHECNQRDEECVTDFVENFVLDQENEDUUEBIVFDNQECVTDFV

Key 1=3, Key 2=21: DMSDLMCUHVVSQTUHIMMRMZYVTEHIUVVTMKUWMVEWMCUHVEVULLVSZMWUEHVTMKUWM

Key 1=3, Key 2=22: UDJUCDTLYMMJHKLYZDDIDQPMKVYZLMMKDBLNDMVNDTLYMVMLCCMJQDNLVYMKDBLND

Key 1=3, Key 2=23: LUALTUKCPDDAYBCPQUUZUHGDBMPQCDDBUSCEUDMEUKCPDMDCTTDAHUECMPDBUSCEU

Key 1=3, Key 2=24: CLRCKLBTGUURPSTGHLLQLYXUSDGHTUUSLJTVLUDVLBTGUDUTKKURYLVTDGUSLJTVL

Key 1=3, Key 2=25: TCITBCSKXLLIGJKXYCCHCPOLJUXYKLLJCAKMCLUMCSKXLULKBBLIPCMKUXLJCAKMC

Key 1=5, Key 2=0: GBPGQBVLYWWPTALYJBBEBODWARYJLWWABFLHBWRHBVLYWRWLQQWPOBHLRYWABFLHB

Key 1=5, Key 2=1: LGULVGAQDBBUYFQDOGGJGTIBFWDOQBBFGKQMGBWMGAQDBWBQVVBUTGMQWDBFGKQMG

Key 1=5, Key 2=2: QLZQALFVIGGZDKVITLLOLYNGKBITVGGKLPVRLGBRLFVIGBGVAAGZYLRVBIGKLPVRL

Key 1=5, Key 2=3: VQEVFQKANLLEIPANYQQTQDSLPGNYALLPQUAWQLGWQKANLGLAFFLEDQWAGNLPQUAWQ

Key 1=5, Key 2=4: AVJAKVPFSQQJNUFSDVVYVIXQULSDFQQUVZFBVQLBVPFSQLQFKKQJIVBFLSQUVZFBV

Key 1=5, Key 2=5: FAOFPAUKXVVOSZKXIAADANCVZQXIKVVZAEKGAVQGAUKXVQVKPPVONAGKQXVZAEKGA

Key 1=5, Key 2=6: KFTKUFZPCAATXEPCNFFIFSHAEVCNPAAEFJPLFAVLFZPCAVAPUUATSFLPVCAEFJPLF

Key 1=5, Key 2=7: PKYPZKEUHFFYCJUHSKKNKXMFJAHSUFFJKOUQKFAQKEUHFAFUZZFYXKQUAHFJKOUQK

Key 1=5, Key 2=8: UPDUEPJZMKKDHOZMXPPSPCRKOFMXZKKOPTZVPKFVPJZMKFKZEEKDCPVZFMKOPTZVP

Key 1=5, Key 2=9: ZUIZJUOERPPIMTERCUUXUHWPTKRCEPPTUYEAUPKAUOERPKPEJJPIHUAEKRPTUYEAU

Key 1=5, Key 2=10: EZNEOZTJWUUNRYJWHZZCZMBUYPWHJUUYZDJFZUPFZTJWUPUJOOUNMZFJPWUYZDJFZ

Key 1=5, Key 2=11: JESJTEYOBZZSWDOBMEEHERGZDUBMOZZDEIOKEZUKEYOBZUZOTTZSREKOUBZDEIOKE

Key 1=5, Key 2=12: OJXOYJDTGEEXBITGRJJMJWLEIZGRTEEIJNTPJEZPJDTGEZETYYEXWJPTZGEIJNTPJ

Key 1=5, Key 2=13: TOCTDOIYLJJCGNYLWOOROBQJNELWYJJNOSYUOJEUOIYLJEJYDDJCBOUYELJNOSYUO

Key 1=5, Key 2=14: YTHYITNDQOOHLSDQBTTWTGVOSJQBDOOSTXDZTOJZTNDQOJODIIOHGTZDJQOSTXDZT

Key 1=5, Key 2=15: DYMDNYSIVTTMQXIVGYYBYLATXOVGITTXYCIEYTOEYSIVTOTINNTMLYEIOVTXYCIEY

Key 1=5, Key 2=16: IDRISDXNAYYRVCNALDDGDQFYCTALNYYCDHNJDYTJDXNAYTYNSSYRQDJNTAYCDHNJD

Key 1=5, Key 2=17: NIWNXICSFDDWAHSFQIILIVKDHYFQSDDHIMSOIDYOICSFDYDSXXDWVIOSYFDHIMSOI

Key 1=5, Key 2=18: SNBSCNHXKIIBFMXKVNNQNAPIMDKVXIIMNRXTNIDTNHXKIDIXCCIBANTXDKIMNRXTN

Key 1=5, Key 2=19: XSGXHSMCPNNGKRCPASSVSFUNRIPACNNRSWCYSNIYSMCPNINCHHNGFSYCIPNRSWCYS

Key 1=5, Key 2=20: CXLCMXRHUSSLPWHUFXXAXKZSWNUFHSSWXBHDXSNDXRHUSNSHMMSLKXDHNUSWXBHDX

Key 1=5, Key 2=21: HCQHRCWMZXXQUBMZKCCFCPEXBSZKMXXBCGMICXSICWMZXSXMRRXQPCIMSZXBCGMIC

Key 1=5, Key 2=22: MHVMWHBRECCVZGREPHHKHUJCGXEPRCCGHLRNHCXNHBRECXCRWWCVUHNRXECGHLRNH

Key 1=5, Key 2=23: RMARBMGWJHHAELWJUMMPMZOHLCJUWHHLMQWSMHCSMGWJHCHWBBHAZMSWCJHLMQWSM

Key 1=5, Key 2=24: WRFWGRLBOMMFJQBOZRRURETMQHOZBMMQRVBXRMHXRLBOMHMBGGMFERXBHOMQRVBXR

Key 1=5, Key 2=25: BWKBLWQGTRRKOVGTEWWZWJYRVMTEGRRVWAGCWRMCWQGTRMRGLLRKJWCGMTRVWAGCW

Key 1=7, Key 2=0: IXHIEXPTGMMHVATGZXXOXKRMABGZTMMAXLTFXMBFXPTGMBMTEEMHKXFTBGMAXLTFX

Key 1=7, Key 2=1: TISTPIAERXXSGLERKIIZIVCXLMRKEXXLIWEQIXMQIAERXMXEPPXSVIQEMRXLIWEQI

Key 1=7, Key 2=2: ETDEATLPCIIDRWPCVTTKTGNIWXCVPIIWTHPBTIXBTLPCIXIPAAIDGTBPXCIWTHPBT

Key 1=7, Key 2=3: PEOPLEWANTTOCHANGEEVERYTHINGATTHESAMETIMEWANTITALLTOREMAINTHESAME

Key 1=7, Key 2=4: APZAWPHLYEEZNSLYRPPGPCJESTYRLEESPDLXPETXPHLYETELWWEZCPXLTYESPDLXP

Key 1=7, Key 2=5: LAKLHASWJPPKYDWJCAARANUPDEJCWPPDAOWIAPEIASWJPEPWHHPKNAIWEJPDAOWIA

Key 1=7, Key 2=6: WLVWSLDHUAAVJOHUNLLCLYFAOPUNHAAOLZHTLAPTLDHUAPAHSSAVYLTHPUAOLZHTL

Key 1=7, Key 2=7: HWGHDWOSFLLGUZSFYWWNWJQLZAFYSLLZWKSEWLAEWOSFLALSDDLGJWESAFLZWKSEW

Key 1=7, Key 2=8: SHRSOHZDQWWRFKDQJHHYHUBWKLQJDWWKHVDPHWLPHZDQWLWDOOWRUHPDLQWKHVDPH

Key 1=7, Key 2=9: DSCDZSKOBHHCQVOBUSSJSFMHVWBUOHHVSGOASHWASKOBHWHOZZHCFSAOWBHVSGOAS

Key 1=7, Key 2=10: ODNOKDVZMSSNBGZMFDDUDQXSGHMFZSSGDRZLDSHLDVZMSHSZKKSNQDLZHMSGDRZLD

Key 1=7, Key 2=11: ZOYZVOGKXDDYMRKXQOOFOBIDRSXQKDDROCKWODSWOGKXDSDKVVDYBOWKSXDROCKWO

Key 1=7, Key 2=12: KZJKGZRVIOOJXCVIBZZQZMTOCDIBVOOCZNVHZODHZRVIODOVGGOJMZHVDIOCZNVHZ

Key 1=7, Key 2=13: VKUVRKCGTZZUINGTMKKBKXEZNOTMGZZNKYGSKZOSKCGTZOZGRRZUXKSGOTZNKYGSK

Key 1=7, Key 2=14: GVFGCVNREKKFTYREXVVMVIPKYZEXRKKYVJRDVKZDVNREKZKRCCKFIVDRZEKYVJRDV

Key 1=7, Key 2=15: RGQRNGYCPVVQEJCPIGGXGTAVJKPICVVJGUCOGVKOGYCPVKVCNNVQTGOCKPVJGUCOG

Key 1=7, Key 2=16: CRBCYRJNAGGBPUNATRRIRELGUVATNGGURFNZRGVZRJNAGVGNYYGBERZNVAGURFNZR

Key 1=7, Key 2=17: NCMNJCUYLRRMAFYLECCTCPWRFGLEYRRFCQYKCRGKCUYLRGRYJJRMPCKYGLRFCQYKC

Key 1=7, Key 2=18: YNXYUNFJWCCXLQJWPNNENAHCQRWPJCCQNBJVNCRVNFJWCRCJUUCXANVJRWCQNBJVN

Key 1=7, Key 2=19: JYIJFYQUHNNIWBUHAYYPYLSNBCHAUNNBYMUGYNCGYQUHNCNUFFNILYGUCHNBYMUGY

Key 1=7, Key 2=20: UJTUQJBFSYYTHMFSLJJAJWDYMNSLFYYMJXFRJYNRJBFSYNYFQQYTWJRFNSYMJXFRJ

Key 1=7, Key 2=21: FUEFBUMQDJJESXQDWUULUHOJXYDWQJJXUIQCUJYCUMQDJYJQBBJEHUCQYDJXUIQCU

Key 1=7, Key 2=22: QFPQMFXBOUUPDIBOHFFWFSZUIJOHBUUIFTBNFUJNFXBOUJUBMMUPSFNBJOUIFTBNF

Key 1=7, Key 2=23: BQABXQIMZFFAOTMZSQQHQDKFTUZSMFFTQEMYQFUYQIMZFUFMXXFADQYMUZFTQEMYQ

Key 1=7, Key 2=24: MBLMIBTXKQQLZEXKDBBSBOVQEFKDXQQEBPXJBQFJBTXKQFQXIIQLOBJXFKQEBPXJB

Key 1=7, Key 2=25: XMWXTMEIVBBWKPIVOMMDMZGBPQVOIBBPMAIUMBQUMEIVBQBITTBWZMUIQVBPMAIUM

Key 1=9, Key 2=0: MPRMGPDJWSSRZAJWFPPIPCTSAVWFJSSAPXJBPSVBPDJWSVSJGGSRCPBJVWSAPXJBP

Key 1=9, Key 2=1: JMOJDMAGTPPOWXGTCMMFMZQPXSTCGPPXMUGYMPSYMAGTPSPGDDPOZMYGSTPXMUGYM

Key 1=9, Key 2=2: GJLGAJXDQMMLTUDQZJJCJWNMUPQZDMMUJRDVJMPVJXDQMPMDAAMLWJVDPQMUJRDVJ

Key 1=9, Key 2=3: DGIDXGUANJJIQRANWGGZGTKJRMNWAJJRGOASGJMSGUANJMJAXXJITGSAMNJRGOASG

Key 1=9, Key 2=4: ADFAUDRXKGGFNOXKTDDWDQHGOJKTXGGODLXPDGJPDRXKGJGXUUGFQDPXJKGODLXPD

Key 1=9, Key 2=5: XACXRAOUHDDCKLUHQAATANEDLGHQUDDLAIUMADGMAOUHDGDURRDCNAMUGHDLAIUMA

Key 1=9, Key 2=6: UXZUOXLREAAZHIRENXXQXKBAIDENRAAIXFRJXADJXLREADAROOAZKXJRDEAIXFRJX

Key 1=9, Key 2=7: RUWRLUIOBXXWEFOBKUUNUHYXFABKOXXFUCOGUXAGUIOBXAXOLLXWHUGOABXFUCOGU

Key 1=9, Key 2=8: ORTOIRFLYUUTBCLYHRRKREVUCXYHLUUCRZLDRUXDRFLYUXULIIUTERDLXYUCRZLDR

Key 1=9, Key 2=9: LOQLFOCIVRRQYZIVEOOHOBSRZUVEIRRZOWIAORUAOCIVRURIFFRQBOAIUVRZOWIAO

Key 1=9, Key 2=10: ILNICLZFSOONVWFSBLLELYPOWRSBFOOWLTFXLORXLZFSOROFCCONYLXFRSOWLTFXL

Key 1=9, Key 2=11: FIKFZIWCPLLKSTCPYIIBIVMLTOPYCLLTIQCUILOUIWCPLOLCZZLKVIUCOPLTIQCUI

Key 1=9, Key 2=12: CFHCWFTZMIIHPQZMVFFYFSJIQLMVZIIQFNZRFILRFTZMILIZWWIHSFRZLMIQFNZRF

Key 1=9, Key 2=13: ZCEZTCQWJFFEMNWJSCCVCPGFNIJSWFFNCKWOCFIOCQWJFIFWTTFEPCOWIJFNCKWOC

Key 1=9, Key 2=14: WZBWQZNTGCCBJKTGPZZSZMDCKFGPTCCKZHTLZCFLZNTGCFCTQQCBMZLTFGCKZHTLZ

Key 1=9, Key 2=15: TWYTNWKQDZZYGHQDMWWPWJAZHCDMQZZHWEQIWZCIWKQDZCZQNNZYJWIQCDZHWEQIW

Key 1=9, Key 2=16: QTVQKTHNAWWVDENAJTTMTGXWEZAJNWWETBNFTWZFTHNAWZWNKKWVGTFNZAWETBNFT

Key 1=9, Key 2=17: NQSNHQEKXTTSABKXGQQJQDUTBWXGKTTBQYKCQTWCQEKXTWTKHHTSDQCKWXTBQYKCQ

Key 1=9, Key 2=18: KNPKENBHUQQPXYHUDNNGNARQYTUDHQQYNVHZNQTZNBHUQTQHEEQPANZHTUQYNVHZN

Key 1=9, Key 2=19: HKMHBKYERNNMUVERAKKDKXONVQRAENNVKSEWKNQWKYERNQNEBBNMXKWEQRNVKSEWK

Key 1=9, Key 2=20: EHJEYHVBOKKJRSBOXHHAHULKSNOXBKKSHPBTHKNTHVBOKNKBYYKJUHTBNOKSHPBTH

Key 1=9, Key 2=21: BEGBVESYLHHGOPYLUEEXERIHPKLUYHHPEMYQEHKQESYLHKHYVVHGREQYKLHPEMYQE

Key 1=9, Key 2=22: YBDYSBPVIEEDLMVIRBBUBOFEMHIRVEEMBJVNBEHNBPVIEHEVSSEDOBNVHIEMBJVNB

Key 1=9, Key 2=23: VYAVPYMSFBBAIJSFOYYRYLCBJEFOSBBJYGSKYBEKYMSFBEBSPPBALYKSEFBJYGSKY

Key 1=9, Key 2=24: SVXSMVJPCYYXFGPCLVVOVIZYGBCLPYYGVDPHVYBHVJPCYBYPMMYXIVHPBCYGVDPHV

Key 1=9, Key 2=25: PSUPJSGMZVVUCDMZISSLSFWVDYZIMVVDSAMESVYESGMZVYVMJJVUFSEMYZVDSAMES

Key 1=11, Key 2=0: YRVYMRTFSKKVLAFSXRRQREZKADSXFKKARHFPRKDPRTFSKDKFMMKVERPFDSKARHFPR

Key 1=11, Key 2=1: FYCFTYAMZRRCSHMZEYYXYLGRHKZEMRRHYOMWYRKWYAMZRKRMTTRCLYWMKZRHYOMWY

Key 1=11, Key 2=2: MFJMAFHTGYYJZOTGLFFEFSNYORGLTYYOFVTDFYRDFHTGYRYTAAYJSFDTRGYOFVTDF

Key 1=11, Key 2=3: TMQTHMOANFFQGVANSMMLMZUFVYNSAFFVMCAKMFYKMOANFYFAHHFQZMKAYNFVMCAKM

Key 1=11, Key 2=4: ATXAOTVHUMMXNCHUZTTSTGBMCFUZHMMCTJHRTMFRTVHUMFMHOOMXGTRHFUMCTJHRT

Key 1=11, Key 2=5: HAEHVACOBTTEUJOBGAAZANITJMBGOTTJAQOYATMYACOBTMTOVVTENAYOMBTJAQOYA

Key 1=11, Key 2=6: OHLOCHJVIAALBQVINHHGHUPAQTINVAAQHXVFHATFHJVIATAVCCALUHFVTIAQHXVFH

Key 1=11, Key 2=7: VOSVJOQCPHHSIXCPUOONOBWHXAPUCHHXOECMOHAMOQCPHAHCJJHSBOMCAPHXOECMO

Key 1=11, Key 2=8: CVZCQVXJWOOZPEJWBVVUVIDOEHWBJOOEVLJTVOHTVXJWOHOJQQOZIVTJHWOEVLJTV

Key 1=11, Key 2=9: JCGJXCEQDVVGWLQDICCBCPKVLODIQVVLCSQACVOACEQDVOVQXXVGPCAQODVLCSQAC

Key 1=11, Key 2=10: QJNQEJLXKCCNDSXKPJJIJWRCSVKPXCCSJZXHJCVHJLXKCVCXEECNWJHXVKCSJZXHJ

Key 1=11, Key 2=11: XQUXLQSERJJUKZERWQQPQDYJZCRWEJJZQGEOQJCOQSERJCJELLJUDQOECRJZQGEOQ

Key 1=11, Key 2=12: EXBESXZLYQQBRGLYDXXWXKFQGJYDLQQGXNLVXQJVXZLYQJQLSSQBKXVLJYQGXNLVX

Key 1=11, Key 2=13: LEILZEGSFXXIYNSFKEEDERMXNQFKSXXNEUSCEXQCEGSFXQXSZZXIRECSQFXNEUSCE

Key 1=11, Key 2=14: SLPSGLNZMEEPFUZMRLLKLYTEUXMRZEEULBZJLEXJLNZMEXEZGGEPYLJZXMEULBZJL

Key 1=11, Key 2=15: ZSWZNSUGTLLWMBGTYSSRSFALBETYGLLBSIGQSLEQSUGTLELGNNLWFSQGETLBSIGQS

Key 1=11, Key 2=16: GZDGUZBNASSDTINAFZZYZMHSILAFNSSIZPNXZSLXZBNASLSNUUSDMZXNLASIZPNXZ

Key 1=11, Key 2=17: NGKNBGIUHZZKAPUHMGGFGTOZPSHMUZZPGWUEGZSEGIUHZSZUBBZKTGEUSHZPGWUEG

Key 1=11, Key 2=18: UNRUINPBOGGRHWBOTNNMNAVGWZOTBGGWNDBLNGZLNPBOGZGBIIGRANLBZOGWNDBLN

Key 1=11, Key 2=19: BUYBPUWIVNNYODIVAUUTUHCNDGVAINNDUKISUNGSUWIVNGNIPPNYHUSIGVNDUKISU

Key 1=11, Key 2=20: IBFIWBDPCUUFVKPCHBBABOJUKNCHPUUKBRPZBUNZBDPCUNUPWWUFOBZPNCUKBRPZB

Key 1=11, Key 2=21: PIMPDIKWJBBMCRWJOIIHIVQBRUJOWBBRIYWGIBUGIKWJBUBWDDBMVIGWUJBRIYWGI

Key 1=11, Key 2=22: WPTWKPRDQIITJYDQVPPOPCXIYBQVDIIYPFDNPIBNPRDQIBIDKKITCPNDBQIYPFDNP

Key 1=11, Key 2=23: DWADRWYKXPPAQFKXCWWVWJEPFIXCKPPFWMKUWPIUWYKXPIPKRRPAJWUKIXPFWMKUW

Key 1=11, Key 2=24: KDHKYDFREWWHXMREJDDCDQLWMPEJRWWMDTRBDWPBDFREWPWRYYWHQDBRPEWMDTRBD

Key 1=11, Key 2=25: RKORFKMYLDDOETYLQKKJKXSDTWLQYDDTKAYIKDWIKMYLDWDYFFDOXKIYWLDTKAYIK

Key 1=15, Key 2=0: CJFCOJHVIQQFPAVIDJJKJWBQAXIDVQQAJTVLJQXLJHVIQXQVOOQFWJLVXIQAJTVLJ

Key 1=15, Key 2=1: VCYVHCAOBJJYITOBWCCDCPUJTQBWOJJTCMOECJQECAOBJQJOHHJYPCEOQBJTCMOEC

Key 1=15, Key 2=2: OVROAVTHUCCRBMHUPVVWVINCMJUPHCCMVFHXVCJXVTHUCJCHAACRIVXHJUCMVFHXV

Key 1=15, Key 2=3: HOKHTOMANVVKUFANIOOPOBGVFCNIAVVFOYAQOVCQOMANVCVATTVKBOQACNVFOYAQO

Key 1=15, Key 2=4: AHDAMHFTGOODNYTGBHHIHUZOYVGBTOOYHRTJHOVJHFTGOVOTMMODUHJTVGOYHRTJH

Key 1=15, Key 2=5: TAWTFAYMZHHWGRMZUAABANSHROZUMHHRAKMCAHOCAYMZHOHMFFHWNACMOZHRAKMCA

Key 1=15, Key 2=6: MTPMYTRFSAAPZKFSNTTUTGLAKHSNFAAKTDFVTAHVTRFSAHAFYYAPGTVFHSAKTDFVT

Key 1=15, Key 2=7: FMIFRMKYLTTISDYLGMMNMZETDALGYTTDMWYOMTAOMKYLTATYRRTIZMOYALTDMWYOM

Key 1=15, Key 2=8: YFBYKFDREMMBLWREZFFGFSXMWTEZRMMWFPRHFMTHFDREMTMRKKMBSFHRTEMWFPRHF

Key 1=15, Key 2=9: RYURDYWKXFFUEPKXSYYZYLQFPMXSKFFPYIKAYFMAYWKXFMFKDDFULYAKMXFPYIKAY

Key 1=15, Key 2=10: KRNKWRPDQYYNXIDQLRRSREJYIFQLDYYIRBDTRYFTRPDQYFYDWWYNERTDFQYIRBDTR

Key 1=15, Key 2=11: DKGDPKIWJRRGQBWJEKKLKXCRBYJEWRRBKUWMKRYMKIWJRYRWPPRGXKMWYJRBKUWMK

Key 1=15, Key 2=12: WDZWIDBPCKKZJUPCXDDEDQVKURCXPKKUDNPFDKRFDBPCKRKPIIKZQDFPRCKUDNPFD

Key 1=15, Key 2=13: PWSPBWUIVDDSCNIVQWWXWJODNKVQIDDNWGIYWDKYWUIVDKDIBBDSJWYIKVDNWGIYW

Key 1=15, Key 2=14: IPLIUPNBOWWLVGBOJPPQPCHWGDOJBWWGPZBRPWDRPNBOWDWBUUWLCPRBDOWGPZBRP

Key 1=15, Key 2=15: BIEBNIGUHPPEOZUHCIIJIVAPZWHCUPPZISUKIPWKIGUHPWPUNNPEVIKUWHPZISUKI

Key 1=15, Key 2=16: UBXUGBZNAIIXHSNAVBBCBOTISPAVNIISBLNDBIPDBZNAIPINGGIXOBDNPAISBLNDB

Key 1=15, Key 2=17: NUQNZUSGTBBQALGTOUUVUHMBLITOGBBLUEGWUBIWUSGTBIBGZZBQHUWGITBLUEGWU

Key 1=15, Key 2=18: GNJGSNLZMUUJTEZMHNNONAFUEBMHZUUENXZPNUBPNLZMUBUZSSUJANPZBMUENXZPN

Key 1=15, Key 2=19: ZGCZLGESFNNCMXSFAGGHGTYNXUFASNNXGQSIGNUIGESFNUNSLLNCTGISUFNXGQSIG

Key 1=15, Key 2=20: SZVSEZXLYGGVFQLYTZZAZMRGQNYTLGGQZJLBZGNBZXLYGNGLEEGVMZBLNYGQZJLBZ

Key 1=15, Key 2=21: LSOLXSQERZZOYJERMSSTSFKZJGRMEZZJSCEUSZGUSQERZGZEXXZOFSUEGRZJSCEUS

Key 1=15, Key 2=22: ELHEQLJXKSSHRCXKFLLMLYDSCZKFXSSCLVXNLSZNLJXKSZSXQQSHYLNXZKSCLVXNL

Key 1=15, Key 2=23: XEAXJECQDLLAKVQDYEEFERWLVSDYQLLVEOQGELSGECQDLSLQJJLAREGQSDLVEOQGE

Key 1=15, Key 2=24: QXTQCXVJWEETDOJWRXXYXKPEOLWRJEEOXHJZXELZXVJWELEJCCETKXZJLWEOXHJZX

Key 1=15, Key 2=25: JQMJVQOCPXXMWHCPKQQRQDIXHEPKCXXHQACSQXESQOCPXEXCVVXMDQSCEPXHQACSQ

Key 1=17, Key 2=0: OLJOULXREIIJBAREVLLSLYHIAFEVRIIALDRZLIFZLXREIFIRUUIJYLZRFEIALDRZL

Key 1=17, Key 2=1: ROMRXOAUHLLMEDUHYOOVOBKLDIHYULLDOGUCOLICOAUHLILUXXLMBOCUIHLDOGUCO

Key 1=17, Key 2=2: URPUARDXKOOPHGXKBRRYRENOGLKBXOOGRJXFROLFRDXKOLOXAAOPERFXLKOGRJXFR

Key 1=17, Key 2=3: XUSXDUGANRRSKJANEUUBUHQRJONEARRJUMAIUROIUGANRORADDRSHUIAONRJUMAIU

Key 1=17, Key 2=4: AXVAGXJDQUUVNMDQHXXEXKTUMRQHDUUMXPDLXURLXJDQURUDGGUVKXLDRQUMXPDLX

Key 1=17, Key 2=5: DAYDJAMGTXXYQPGTKAAHANWXPUTKGXXPASGOAXUOAMGTXUXGJJXYNAOGUTXPASGOA

Key 1=17, Key 2=6: GDBGMDPJWAABTSJWNDDKDQZASXWNJAASDVJRDAXRDPJWAXAJMMABQDRJXWASDVJRD

Key 1=17, Key 2=7: JGEJPGSMZDDEWVMZQGGNGTCDVAZQMDDVGYMUGDAUGSMZDADMPPDETGUMAZDVGYMUG

Key 1=17, Key 2=8: MJHMSJVPCGGHZYPCTJJQJWFGYDCTPGGYJBPXJGDXJVPCGDGPSSGHWJXPDCGYJBPXJ

Key 1=17, Key 2=9: PMKPVMYSFJJKCBSFWMMTMZIJBGFWSJJBMESAMJGAMYSFJGJSVVJKZMASGFJBMESAM

Key 1=17, Key 2=10: SPNSYPBVIMMNFEVIZPPWPCLMEJIZVMMEPHVDPMJDPBVIMJMVYYMNCPDVJIMEPHVDP

Key 1=17, Key 2=11: VSQVBSEYLPPQIHYLCSSZSFOPHMLCYPPHSKYGSPMGSEYLPMPYBBPQFSGYMLPHSKYGS

Key 1=17, Key 2=12: YVTYEVHBOSSTLKBOFVVCVIRSKPOFBSSKVNBJVSPJVHBOSPSBEESTIVJBPOSKVNBJV

Key 1=17, Key 2=13: BYWBHYKERVVWONERIYYFYLUVNSRIEVVNYQEMYVSMYKERVSVEHHVWLYMESRVNYQEMY

Key 1=17, Key 2=14: EBZEKBNHUYYZRQHULBBIBOXYQVULHYYQBTHPBYVPBNHUYVYHKKYZOBPHVUYQBTHPB

Key 1=17, Key 2=15: HECHNEQKXBBCUTKXOEELERABTYXOKBBTEWKSEBYSEQKXBYBKNNBCRESKYXBTEWKSE

Key 1=17, Key 2=16: KHFKQHTNAEEFXWNARHHOHUDEWBARNEEWHZNVHEBVHTNAEBENQQEFUHVNBAEWHZNVH

Key 1=17, Key 2=17: NKINTKWQDHHIAZQDUKKRKXGHZEDUQHHZKCQYKHEYKWQDHEHQTTHIXKYQEDHZKCQYK

Key 1=17, Key 2=18: QNLQWNZTGKKLDCTGXNNUNAJKCHGXTKKCNFTBNKHBNZTGKHKTWWKLANBTHGKCNFTBN

Key 1=17, Key 2=19: TQOTZQCWJNNOGFWJAQQXQDMNFKJAWNNFQIWEQNKEQCWJNKNWZZNODQEWKJNFQIWEQ

Key 1=17, Key 2=20: WTRWCTFZMQQRJIZMDTTATGPQINMDZQQITLZHTQNHTFZMQNQZCCQRGTHZNMQITLZHT

Key 1=17, Key 2=21: ZWUZFWICPTTUMLCPGWWDWJSTLQPGCTTLWOCKWTQKWICPTQTCFFTUJWKCQPTLWOCKW

Key 1=17, Key 2=22: CZXCIZLFSWWXPOFSJZZGZMVWOTSJFWWOZRFNZWTNZLFSWTWFIIWXMZNFTSWOZRFNZ

Key 1=17, Key 2=23: FCAFLCOIVZZASRIVMCCJCPYZRWVMIZZRCUIQCZWQCOIVZWZILLZAPCQIWVZRCUIQC

Key 1=17, Key 2=24: IFDIOFRLYCCDVULYPFFMFSBCUZYPLCCUFXLTFCZTFRLYCZCLOOCDSFTLZYCUFXLTF

Key 1=17, Key 2=25: LIGLRIUOBFFGYXOBSIIPIVEFXCBSOFFXIAOWIFCWIUOBFCFORRFGVIWOCBFXIAOWI

Key 1=19, Key 2=0: SDTSWDLHUOOTFAHUBDDMDQJOAZUBHOOADPHVDOZVDLHUOZOHWWOTQDVHZUOADPHVD

Key 1=19, Key 2=1: HSIHLSAWJDDIUPWJQSSBSFYDPOJQWDDPSEWKSDOKSAWJDODWLLDIFSKWOJDPSEWKS

Key 1=19, Key 2=2: WHXWAHPLYSSXJELYFHHQHUNSEDYFLSSEHTLZHSDZHPLYSDSLAASXUHZLDYSEHTLZH

Key 1=19, Key 2=3: LWMLPWEANHHMYTANUWWFWJCHTSNUAHHTWIAOWHSOWEANHSHAPPHMJWOASNHTWIAOW

Key 1=19, Key 2=4: ALBAELTPCWWBNIPCJLLULYRWIHCJPWWILXPDLWHDLTPCWHWPEEWBYLDPHCWILXPDL

Key 1=19, Key 2=5: PAQPTAIERLLQCXERYAAJANGLXWRYELLXAMESALWSAIERLWLETTLQNASEWRLXAMESA

Key 1=19, Key 2=6: EPFEIPXTGAAFRMTGNPPYPCVAMLGNTAAMPBTHPALHPXTGALATIIAFCPHTLGAMPBTHP

Key 1=19, Key 2=7: TEUTXEMIVPPUGBIVCEENERKPBAVCIPPBEQIWEPAWEMIVPAPIXXPUREWIAVPBEQIWE

Key 1=19, Key 2=8: ITJIMTBXKEEJVQXKRTTCTGZEQPKRXEEQTFXLTEPLTBXKEPEXMMEJGTLXPKEQTFXLT

Key 1=19, Key 2=9: XIYXBIQMZTTYKFMZGIIRIVOTFEZGMTTFIUMAITEAIQMZTETMBBTYVIAMEZTFIUMAI

Key 1=19, Key 2=10: MXNMQXFBOIINZUBOVXXGXKDIUTOVBIIUXJBPXITPXFBOITIBQQINKXPBTOIUXJBPX

Key 1=19, Key 2=11: BMCBFMUQDXXCOJQDKMMVMZSXJIDKQXXJMYQEMXIEMUQDXIXQFFXCZMEQIDXJMYQEM

Key 1=19, Key 2=12: QBRQUBJFSMMRDYFSZBBKBOHMYXSZFMMYBNFTBMXTBJFSMXMFUUMROBTFXSMYBNFTB

Key 1=19, Key 2=13: FQGFJQYUHBBGSNUHOQQZQDWBNMHOUBBNQCUIQBMIQYUHBMBUJJBGDQIUMHBNQCUIQ

Key 1=19, Key 2=14: UFVUYFNJWQQVHCJWDFFOFSLQCBWDJQQCFRJXFQBXFNJWQBQJYYQVSFXJBWQCFRJXF

Key 1=19, Key 2=15: JUKJNUCYLFFKWRYLSUUDUHAFRQLSYFFRUGYMUFQMUCYLFQFYNNFKHUMYQLFRUGYMU

Key 1=19, Key 2=16: YJZYCJRNAUUZLGNAHJJSJWPUGFAHNUUGJVNBJUFBJRNAUFUNCCUZWJBNFAUGJVNBJ

Key 1=19, Key 2=17: NYONRYGCPJJOAVCPWYYHYLEJVUPWCJJVYKCQYJUQYGCPJUJCRRJOLYQCUPJVYKCQY

Key 1=19, Key 2=18: CNDCGNVREYYDPKRELNNWNATYKJELRYYKNZRFNYJFNVREYJYRGGYDANFRJEYKNZRFN

Key 1=19, Key 2=19: RCSRVCKGTNNSEZGTACCLCPINZYTAGNNZCOGUCNYUCKGTNYNGVVNSPCUGYTNZCOGUC

Key 1=19, Key 2=20: GRHGKRZVICCHTOVIPRRAREXCONIPVCCORDVJRCNJRZVICNCVKKCHERJVNICORDVJR

Key 1=19, Key 2=21: VGWVZGOKXRRWIDKXEGGPGTMRDCXEKRRDGSKYGRCYGOKXRCRKZZRWTGYKCXRDGSKYG

Key 1=19, Key 2=22: KVLKOVDZMGGLXSZMTVVEVIBGSRMTZGGSVHZNVGRNVDZMGRGZOOGLIVNZRMGSVHZNV

Key 1=19, Key 2=23: ZKAZDKSOBVVAMHOBIKKTKXQVHGBIOVVHKWOCKVGCKSOBVGVODDVAXKCOGBVHKWOCK

Key 1=19, Key 2=24: OZPOSZHDQKKPBWDQXZZIZMFKWVQXDKKWZLDRZKVRZHDQKVKDSSKPMZRDVQKWZLDRZ

Key 1=19, Key 2=25: DOEDHOWSFZZEQLSFMOOXOBUZLKFMSZZLOASGOZKGOWSFZKZSHHZEBOGSKFZLOASGO

Key 1=21, Key 2=0: UZLUKZFPCEELHAPCRZZWZMXEAJCRPEEAZVPTZEJTZFPCEJEPKKELMZTPJCEAZVPTZ

Key 1=21, Key 2=1: PUGPFUAKXZZGCVKXMUURUHSZVEXMKZZVUQKOUZEOUAKXZEZKFFZGHUOKEXZVUQKOU

Key 1=21, Key 2=2: KPBKAPVFSUUBXQFSHPPMPCNUQZSHFUUQPLFJPUZJPVFSUZUFAAUBCPJFZSUQPLFJP

Key 1=21, Key 2=3: FKWFVKQANPPWSLANCKKHKXIPLUNCAPPLKGAEKPUEKQANPUPAVVPWXKEAUNPLKGAEK

Key 1=21, Key 2=4: AFRAQFLVIKKRNGVIXFFCFSDKGPIXVKKGFBVZFKPZFLVIKPKVQQKRSFZVPIKGFBVZF

Key 1=21, Key 2=5: VAMVLAGQDFFMIBQDSAAXANYFBKDSQFFBAWQUAFKUAGQDFKFQLLFMNAUQKDFBAWQUA

Key 1=21, Key 2=6: QVHQGVBLYAAHDWLYNVVSVITAWFYNLAAWVRLPVAFPVBLYAFALGGAHIVPLFYAWVRLPV

Key 1=21, Key 2=7: LQCLBQWGTVVCYRGTIQQNQDOVRATIGVVRQMGKQVAKQWGTVAVGBBVCDQKGATVRQMGKQ

Key 1=21, Key 2=8: GLXGWLRBOQQXTMBODLLILYJQMVODBQQMLHBFLQVFLRBOQVQBWWQXYLFBVOQMLHBFL

Key 1=21, Key 2=9: BGSBRGMWJLLSOHWJYGGDGTELHQJYWLLHGCWAGLQAGMWJLQLWRRLSTGAWQJLHGCWAG

Key 1=21, Key 2=10: WBNWMBHREGGNJCRETBBYBOZGCLETRGGCBXRVBGLVBHREGLGRMMGNOBVRLEGCBXRVB

Key 1=21, Key 2=11: RWIRHWCMZBBIEXMZOWWTWJUBXGZOMBBXWSMQWBGQWCMZBGBMHHBIJWQMGZBXWSMQW

Key 1=21, Key 2=12: MRDMCRXHUWWDZSHUJRROREPWSBUJHWWSRNHLRWBLRXHUWBWHCCWDERLHBUWSRNHLR

Key 1=21, Key 2=13: HMYHXMSCPRRYUNCPEMMJMZKRNWPECRRNMICGMRWGMSCPRWRCXXRYZMGCWPRNMICGM

Key 1=21, Key 2=14: CHTCSHNXKMMTPIXKZHHEHUFMIRKZXMMIHDXBHMRBHNXKMRMXSSMTUHBXRKMIHDXBH

Key 1=21, Key 2=15: XCOXNCISFHHOKDSFUCCZCPAHDMFUSHHDCYSWCHMWCISFHMHSNNHOPCWSMFHDCYSWC

Key 1=21, Key 2=16: SXJSIXDNACCJFYNAPXXUXKVCYHAPNCCYXTNRXCHRXDNACHCNIICJKXRNHACYXTNRX

Key 1=21, Key 2=17: NSENDSYIVXXEATIVKSSPSFQXTCVKIXXTSOIMSXCMSYIVXCXIDDXEFSMICVXTSOIMS

Key 1=21, Key 2=18: INZIYNTDQSSZVODQFNNKNALSOXQFDSSONJDHNSXHNTDQSXSDYYSZANHDXQSONJDHN

Key 1=21, Key 2=19: DIUDTIOYLNNUQJYLAIIFIVGNJSLAYNNJIEYCINSCIOYLNSNYTTNUVICYSLNJIEYCI

Key 1=21, Key 2=20: YDPYODJTGIIPLETGVDDADQBIENGVTIIEDZTXDINXDJTGINITOOIPQDXTNGIEDZTXD

Key 1=21, Key 2=21: TYKTJYEOBDDKGZOBQYYVYLWDZIBQODDZYUOSYDISYEOBDIDOJJDKLYSOIBDZYUOSY

Key 1=21, Key 2=22: OTFOETZJWYYFBUJWLTTQTGRYUDWLJYYUTPJNTYDNTZJWYDYJEEYFGTNJDWYUTPJNT

Key 1=21, Key 2=23: JOAJZOUERTTAWPERGOOLOBMTPYRGETTPOKEIOTYIOUERTYTEZZTABOIEYRTPOKEIO

Key 1=21, Key 2=24: EJVEUJPZMOOVRKZMBJJGJWHOKTMBZOOKJFZDJOTDJPZMOTOZUUOVWJDZTMOKJFZDJ

Key 1=21, Key 2=25: ZEQZPEKUHJJQMFUHWEEBERCJFOHWUJJFEAUYEJOYEKUHJOJUPPJQREYUOHJFEAUYE

Key 1=23, Key 2=0: QHBQIHRZMYYBDAZMLHHCHUVYAPMLZYYAHJZXHYPXHRZMYPYZIIYBUHXZPMYAHJZXH

Key 1=23, Key 2=1: ZQKZRQAIVHHKMJIVUQQLQDEHJYVUIHHJQSIGQHYGQAIVHYHIRRHKDQGIYVHJQSIGQ

Key 1=23, Key 2=2: IZTIAZJREQQTVSREDZZUZMNQSHEDRQQSZBRPZQHPZJREQHQRAAQTMZPRHEQSZBRPZ

Key 1=23, Key 2=3: RICRJISANZZCEBANMIIDIVWZBQNMAZZBIKAYIZQYISANZQZAJJZCVIYAQNZBIKAYI

Key 1=23, Key 2=4: ARLASRBJWIILNKJWVRRMREFIKZWVJIIKRTJHRIZHRBJWIZIJSSILERHJZWIKRTJHR

Key 1=23, Key 2=5: JAUJBAKSFRRUWTSFEAAVANORTIFESRRTACSQARIQAKSFRIRSBBRUNAQSIFRTACSQA

Key 1=23, Key 2=6: SJDSKJTBOAADFCBONJJEJWXACRONBAACJLBZJARZJTBOARABKKADWJZBROACJLBZJ

Key 1=23, Key 2=7: BSMBTSCKXJJMOLKXWSSNSFGJLAXWKJJLSUKISJAISCKXJAJKTTJMFSIKAXJLSUKIS

Key 1=23, Key 2=8: KBVKCBLTGSSVXUTGFBBWBOPSUJGFTSSUBDTRBSJRBLTGSJSTCCSVOBRTJGSUBDTRB

Key 1=23, Key 2=9: TKETLKUCPBBEGDCPOKKFKXYBDSPOCBBDKMCAKBSAKUCPBSBCLLBEXKACSPBDKMCAK

Key 1=23, Key 2=10: CTNCUTDLYKKNPMLYXTTOTGHKMBYXLKKMTVLJTKBJTDLYKBKLUUKNGTJLBYKMTVLJT

Key 1=23, Key 2=11: LCWLDCMUHTTWYVUHGCCXCPQTVKHGUTTVCEUSCTKSCMUHTKTUDDTWPCSUKHTVCEUSC

Key 1=23, Key 2=12: ULFUMLVDQCCFHEDQPLLGLYZCETQPDCCELNDBLCTBLVDQCTCDMMCFYLBDTQCELNDBL

Key 1=23, Key 2=13: DUODVUEMZLLOQNMZYUUPUHILNCZYMLLNUWMKULCKUEMZLCLMVVLOHUKMCZLNUWMKU

Key 1=23, Key 2=14: MDXMEDNVIUUXZWVIHDDYDQRUWLIHVUUWDFVTDULTDNVIULUVEEUXQDTVLIUWDFVTD

Key 1=23, Key 2=15: VMGVNMWERDDGIFERQMMHMZADFURQEDDFMOECMDUCMWERDUDENNDGZMCEURDFMOECM

Key 1=23, Key 2=16: EVPEWVFNAMMPRONAZVVQVIJMODAZNMMOVXNLVMDLVFNAMDMNWWMPIVLNDAMOVXNLV

Key 1=23, Key 2=17: NEYNFEOWJVVYAXWJIEEZERSVXMJIWVVXEGWUEVMUEOWJVMVWFFVYREUWMJVXEGWUE

Key 1=23, Key 2=18: WNHWONXFSEEHJGFSRNNINABEGVSRFEEGNPFDNEVDNXFSEVEFOOEHANDFVSEGNPFDN

Key 1=23, Key 2=19: FWQFXWGOBNNQSPOBAWWRWJKNPEBAONNPWYOMWNEMWGOBNENOXXNQJWMOEBNPWYOMW

Key 1=23, Key 2=20: OFZOGFPXKWWZBYXKJFFAFSTWYNKJXWWYFHXVFWNVFPXKWNWXGGWZSFVXNKWYFHXVF

Key 1=23, Key 2=21: XOIXPOYGTFFIKHGTSOOJOBCFHWTSGFFHOQGEOFWEOYGTFWFGPPFIBOEGWTFHOQGEO

Key 1=23, Key 2=22: GXRGYXHPCOORTQPCBXXSXKLOQFCBPOOQXZPNXOFNXHPCOFOPYYORKXNPFCOQXZPNX

Key 1=23, Key 2=23: PGAPHGQYLXXACZYLKGGBGTUXZOLKYXXZGIYWGXOWGQYLXOXYHHXATGWYOLXZGIYWG

Key 1=23, Key 2=24: YPJYQPZHUGGJLIHUTPPKPCDGIXUTHGGIPRHFPGXFPZHUGXGHQQGJCPFHXUGIPRHFP

Key 1=23, Key 2=25: HYSHZYIQDPPSURQDCYYTYLMPRGDCQPPRYAQOYPGOYIQDPGPQZZPSLYOQGDPRYAQOY

Key 1=25, Key 2=0: WVDWYVZXKUUDJAXKHVVGVILUATKHXUUAVBXRVUTRVZXKUTUXYYUDIVRXTKUAVBXRV

Key 1=25, Key 2=1: XWEXZWAYLVVEKBYLIWWHWJMVBULIYVVBWCYSWVUSWAYLVUVYZZVEJWSYULVBWCYSW

Key 1=25, Key 2=2: YXFYAXBZMWWFLCZMJXXIXKNWCVMJZWWCXDZTXWVTXBZMWVWZAAWFKXTZVMWCXDZTX

Key 1=25, Key 2=3: ZYGZBYCANXXGMDANKYYJYLOXDWNKAXXDYEAUYXWUYCANXWXABBXGLYUAWNXDYEAUY

Key 1=25, Key 2=4: AZHACZDBOYYHNEBOLZZKZMPYEXOLBYYEZFBVZYXVZDBOYXYBCCYHMZVBXOYEZFBVZ

Key 1=25, Key 2=5: BAIBDAECPZZIOFCPMAALANQZFYPMCZZFAGCWAZYWAECPZYZCDDZINAWCYPZFAGCWA

Key 1=25, Key 2=6: CBJCEBFDQAAJPGDQNBBMBORAGZQNDAAGBHDXBAZXBFDQAZADEEAJOBXDZQAGBHDXB

Key 1=25, Key 2=7: DCKDFCGERBBKQHEROCCNCPSBHAROEBBHCIEYCBAYCGERBABEFFBKPCYEARBHCIEYC

Key 1=25, Key 2=8: EDLEGDHFSCCLRIFSPDDODQTCIBSPFCCIDJFZDCBZDHFSCBCFGGCLQDZFBSCIDJFZD

Key 1=25, Key 2=9: FEMFHEIGTDDMSJGTQEEPERUDJCTQGDDJEKGAEDCAEIGTDCDGHHDMREAGCTDJEKGAE

Key 1=25, Key 2=10: GFNGIFJHUEENTKHURFFQFSVEKDURHEEKFLHBFEDBFJHUEDEHIIENSFBHDUEKFLHBF

Key 1=25, Key 2=11: HGOHJGKIVFFOULIVSGGRGTWFLEVSIFFLGMICGFECGKIVFEFIJJFOTGCIEVFLGMICG

Key 1=25, Key 2=12: IHPIKHLJWGGPVMJWTHHSHUXGMFWTJGGMHNJDHGFDHLJWGFGJKKGPUHDJFWGMHNJDH

Key 1=25, Key 2=13: JIQJLIMKXHHQWNKXUIITIVYHNGXUKHHNIOKEIHGEIMKXHGHKLLHQVIEKGXHNIOKEI

Key 1=25, Key 2=14: KJRKMJNLYIIRXOLYVJJUJWZIOHYVLIIOJPLFJIHFJNLYIHILMMIRWJFLHYIOJPLFJ

Key 1=25, Key 2=15: LKSLNKOMZJJSYPMZWKKVKXAJPIZWMJJPKQMGKJIGKOMZJIJMNNJSXKGMIZJPKQMGK

Key 1=25, Key 2=16: MLTMOLPNAKKTZQNAXLLWLYBKQJAXNKKQLRNHLKJHLPNAKJKNOOKTYLHNJAKQLRNHL

Key 1=25, Key 2=17: NMUNPMQOBLLUAROBYMMXMZCLRKBYOLLRMSOIMLKIMQOBLKLOPPLUZMIOKBLRMSOIM

Key 1=25, Key 2=18: ONVOQNRPCMMVBSPCZNNYNADMSLCZPMMSNTPJNMLJNRPCMLMPQQMVANJPLCMSNTPJN

Key 1=25, Key 2=19: POWPROSQDNNWCTQDAOOZOBENTMDAQNNTOUQKONMKOSQDNMNQRRNWBOKQMDNTOUQKO

Key 1=25, Key 2=20: QPXQSPTREOOXDUREBPPAPCFOUNEBROOUPVRLPONLPTREONORSSOXCPLRNEOUPVRLP

Key 1=25, Key 2=21: RQYRTQUSFPPYEVSFCQQBQDGPVOFCSPPVQWSMQPOMQUSFPOPSTTPYDQMSOFPVQWSMQ

Key 1=25, Key 2=22: SRZSURVTGQQZFWTGDRRCREHQWPGDTQQWRXTNRQPNRVTGQPQTUUQZERNTPGQWRXTNR

Key 1=25, Key 2=23: TSATVSWUHRRAGXUHESSDSFIRXQHEURRXSYUOSRQOSWUHRQRUVVRAFSOUQHRXSYUOS

Key 1=25, Key 2=24: UTBUWTXVISSBHYVIFTTETGJSYRIFVSSYTZVPTSRPTXVISRSVWWSBGTPVRISYTZVPT

Key 1=25, Key 2=25: VUCVXUYWJTTCIZWJGUUFUHKTZSJGWTTZUAWQUTSQUYWJTSTWXXTCHUQWSJTZUAWQU