DAY-2

1.

Write a High level codefor monoalphabetic cipher is that both sender and receiver must commit thepermuted cipher sequence to memory. A common technique for avoiding this is to use a keyword fromwhich the cipher sequence can be generated.

For example, using the keyword CIPHER, write out the keyword followed by unused letters in normal

order and match this against the plaintext letters:

plain: a b c d e f g h i j k l m n o p q r s t u v w x y z

cipher: C I P H E R A B D F G J K L M N O Q S T U V W X Y Z

programme:

#include<stdio.h>

char monocipher\_encr(char);

char alpha[27][3] = { { 'a', 'C' }, { 'b', 'I' }, { 'c', 'P' }, { 'd', 'H' }, {

'e', 'E' }, { 'f', 'R' }, { 'g', 'A' }, { 'h', 'B' }, { 'i', 'D' }, {

'j', 'F' }, { 'k', 'G' }, { 'l', 'J' }, { 'm', 'K' }, { 'n', 'L' }, {

'o', 'M' }, { 'p', 'N' }, { 'q', 'O' }, { 'r', 'Q' }, { 's', 'S' }, {

't', 'T' }, { 'u', 'U' }, { 'v', 'V' }, { 'w', 'W' }, { 'x', 'X' }, {

'y', 'Y' }, { 'z', 'Z' } };

char str[20];

int main() {

char str[20], str2[20];

int i;

printf("\n Enter String..");

gets(str);

for (i = 0; str[i]; i++) {

str2[i] = monocipher\_encr(str[i]);

}

str2[i] = '\0';

printf("\n Before Decryption..%s", str);

printf("\n After Decryption..%s\n", str2);

}

char monocipher\_encr(char a) {

int i;

for (i = 0; i < 27; i++) {

if (a == alpha[i][0])

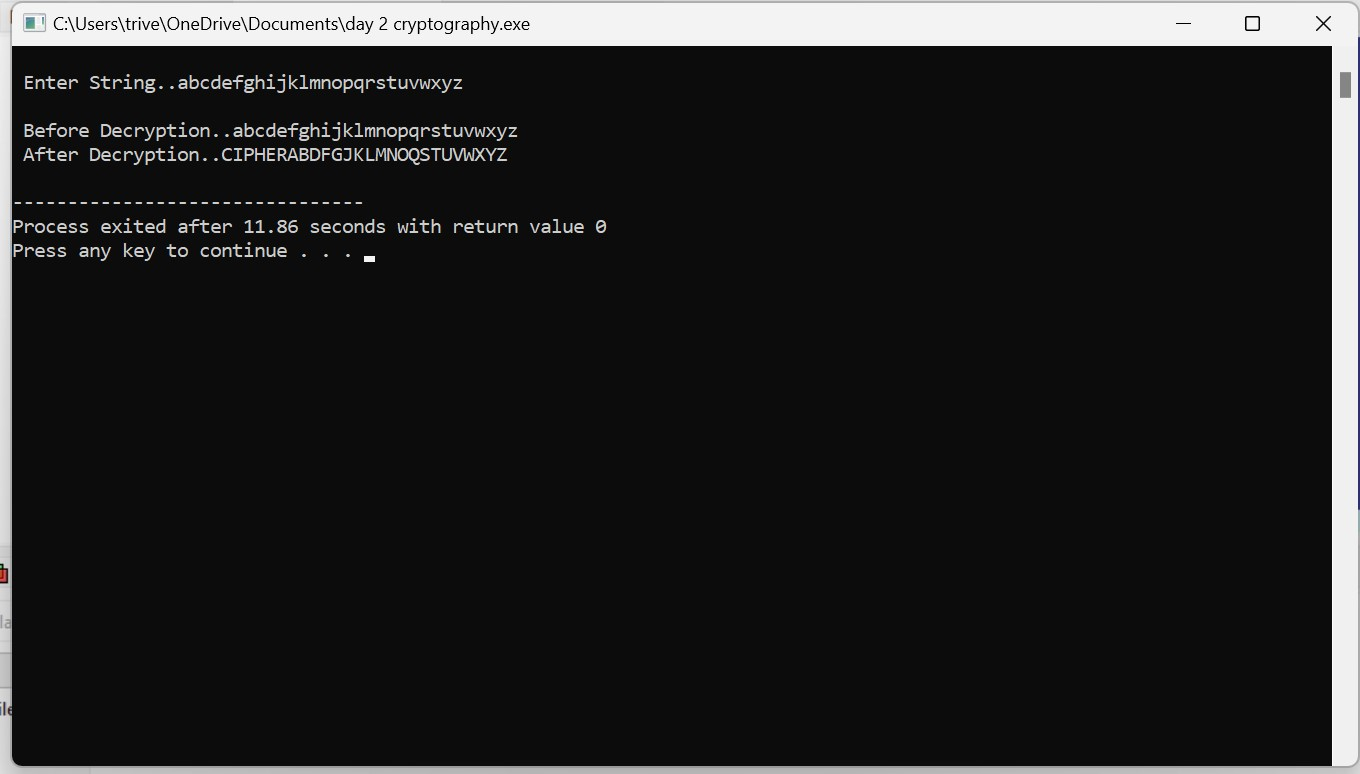
break;

}

return alpha[i][1];

}

Output:



4.Write a High level code for possible keys does the Playfair cipher have? Ignore the fact that some

keys might produce identical encryption results. Express your answer as an approximate power of 2.

a. Now take into account the fact that some Playfair keys produce the same encryption results. How

many effectively unique keys does the Playfair cipher have?

Programme:

#include <stdio.h>

#include <math.h>

int factorial(int n) {

if (n == 0) return 1;

return n \* factorial(n - 1);

}

int main() {

int key\_size = 25;

long double possible\_keys = (long double)factorial(key\_size);

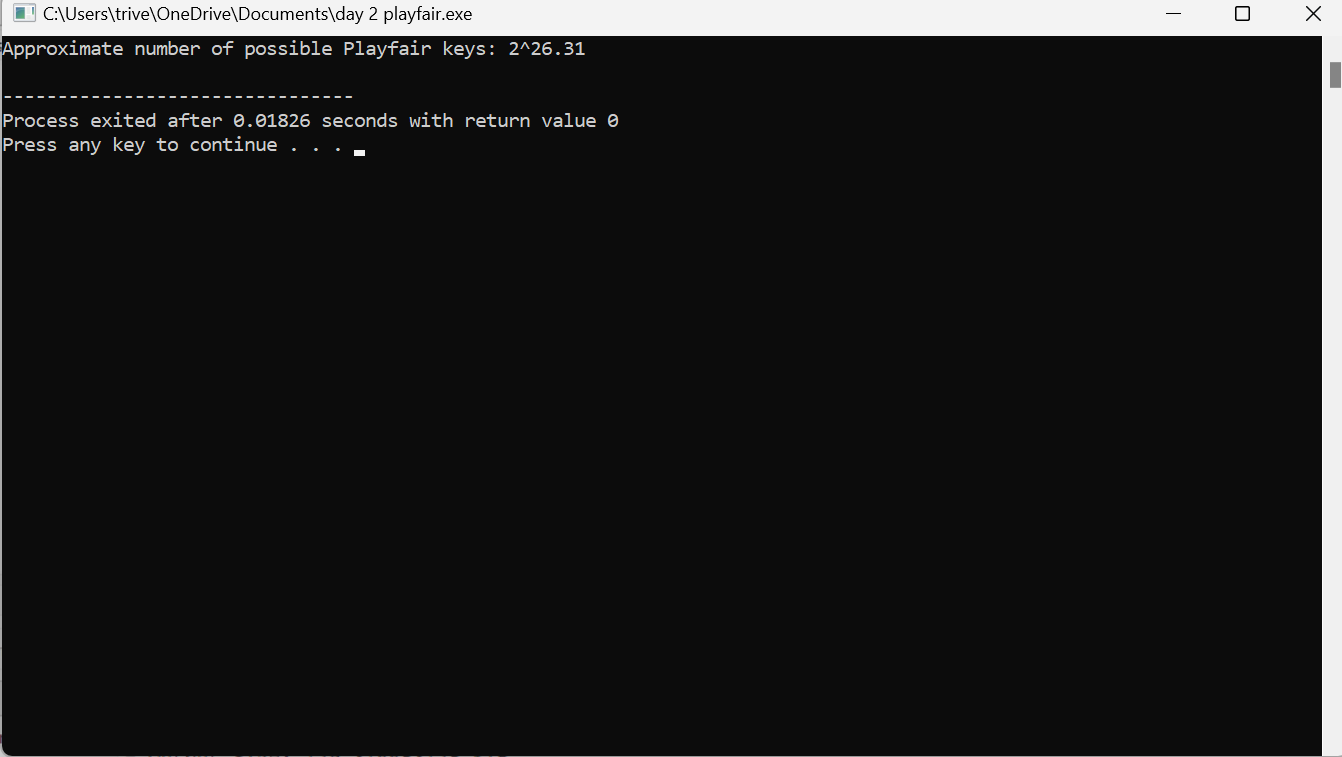
possible\_keys /= key\_size;

double log\_base\_2 = log2(possible\_keys);

printf("Approximate number of possible Playfair keys: 2^%.2lf\n", log\_base\_2);

return 0;

}

Output:

2.

Write a High level codefor PT-109 American patrol boat, under the command of Lieutenant John F.Kennedy, was sunk by a Japanese destroyer, a message was received at an Australian wireless station in Playfair code:

KXJEY UREBE ZWEHE WRYTU HEYFS

KREHE GOYFI WTTTU OLKSY CAJPO

BOTEI ZONTX BYBNT GONEY CUZWR

GDSON SXBOU YWRHE BAAHY USEDQ

Programme:

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void buildPlayfairMatrix(char matrix[5][5], const char\* keyword) {

char used[26] = {0};

int row = 0, col = 0;

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

char letter = tolower(keyword[i]);

if (!used[letter - 'a']) {

matrix[row][col] = letter;

used[letter - 'a'] = 1;

col++;

if (col == 5) {

col = 0;

row++;

}

}

}

for (char letter = 'a'; letter <= 'z'; letter++) {

if (letter != 'j' && !used[letter - 'a']) {

matrix[row][col] = letter;

col++;

if (col == 5) {

col = 0;

row++;

}

}

}

}

void decipherPlayfair(const char matrix[5][5], const char\* ciphertext) {

int len = strlen(ciphertext);

char plaintext[len + 1];

int row1, col1, row2, col2;

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

char char1 = tolower(ciphertext[i]);

char char2 = tolower(ciphertext[i + 1]);

if (char1 == char2) {

char2 = 'x';

i--;

}

for (int row = 0; row < 5; row++) {

for (int col = 0; col < 5; col++) {

if (matrix[row][col] == char1) {

row1 = row;

col1 = col;

}

if (matrix[row][col] == char2) {

row2 = row;

col2 = col;

}

}

}

if (row1 == row2) {

plaintext[i] = matrix[row1][(col1 + 4) % 5];

plaintext[i + 1] = matrix[row2][(col2 + 4) % 5];

} else if (col1 == col2) {

plaintext[i] = matrix[(row1 + 4) % 5][col1];

plaintext[i + 1] = matrix[(row2 + 4) % 5][col2];

} else {

plaintext[i] = matrix[row1][col2];

plaintext[i + 1] = matrix[row2][col1];

}

}

plaintext[len] = '\0';

printf("Deciphered Message: %s\n", plaintext);

}

int main() {

const char\* keyword = "KEYWORD";

const char\* ciphertext = "KXJEY UREBE ZWEHE WRYTU HEYFS KREHE GOYFI WTTTU OLKSY CAJPO BOTEI ZONTX BYBNT GONEY CUZWR GDSON SXBOU YWRHE BAAHY USEDQ";

char matrix[5][5];

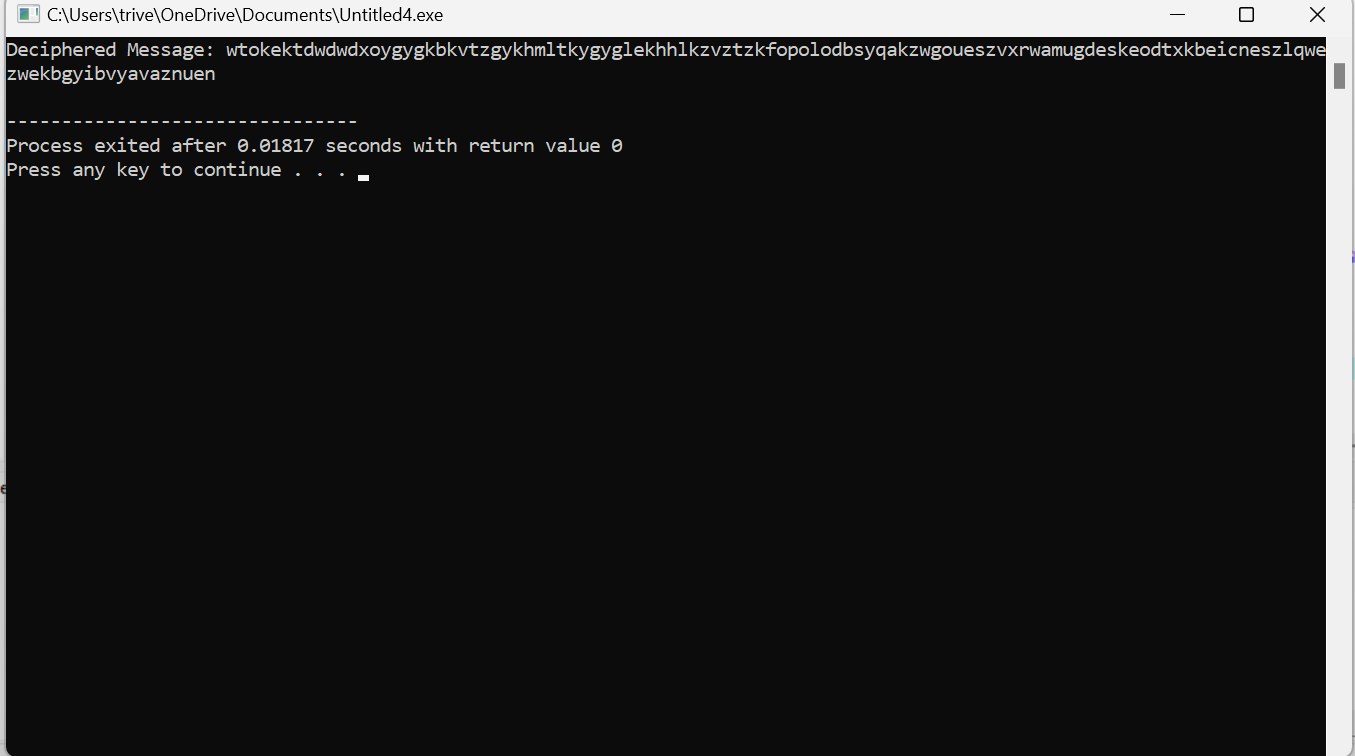
buildPlayfairMatrix(matrix, keyword);

decipherPlayfair(matrix, ciphertext);

return 0;

}

Output:



5.

Write a High level code to Encrypt the message “meet me at the usual place at ten rather than eight

oclock” using the Hill cipher with the key.

9 4

5 7

a. Show your calculations and the result.

b. Show the calculations for the corresponding decryption of the ciphertext to recover the original

plaintext.

Programme:

#include <stdio.h>

#include <string.h>

#define N 2

void encrypt(char message[], int key[N][N]) {

int i, j, k;

int len = strlen(message);

if (len % N != 0) {

printf("Message length is not a multiple of %d. Padding with 'X'.\n", N);

while (len % N != 0) {

message[len++] = 'X';

}

message[len] = '\0';

}

int numericMessage[len];

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

numericMessage[i] = message[i] - 'a';

}

int encryptedMessage[len];

for (i = 0; i < len; i += N) {

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

encryptedMessage[i + j] = 0;

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

encryptedMessage[i + j] += key[j][k] \* numericMessage[i + k];

}

encryptedMessage[i + j] %= 26;

}

}

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

message[i] = (char)(encryptedMessage[i] + 'a');

}

}

void decrypt(char message[], int key[N][N]) {

int det = (key[0][0] \* key[1][1] - key[0][1] \* key[1][0] + 26) % 26;

int detInv;

for (detInv = 1; detInv < 26; detInv++) {

if ((det \* detInv) % 26 == 1) {

break;

}

}

int keyAdj[N][N] = {{key[1][1], -key[0][1]}, {-key[1][0], key[0][0]}};

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

for (int j = 0; j < N; j++) {

keyAdj[i][j] = (keyAdj[i][j] \* detInv + 26) % 26;

}

}

int len = strlen(message);

int numericMessage[len];

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

numericMessage[i] = message[i] - 'a';

}

int decryptedMessage[len];

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

for (int j = 0; j < N; j++) {

decryptedMessage[i + j] = 0;

for (int k = 0; k < N; k++) {

decryptedMessage[i + j] += keyAdj[j][k] \* numericMessage[i + k];

}

decryptedMessage[i + j] %= 26;

}

}

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

message[i] = (char)(decryptedMessage[i] + 'a');

}

}

int main() {

int key[N][N] = {{9, 4}, {5, 7}};

char message[] = "meet me at the usual place at ten rather than eight oclock";

encrypt(message, key);

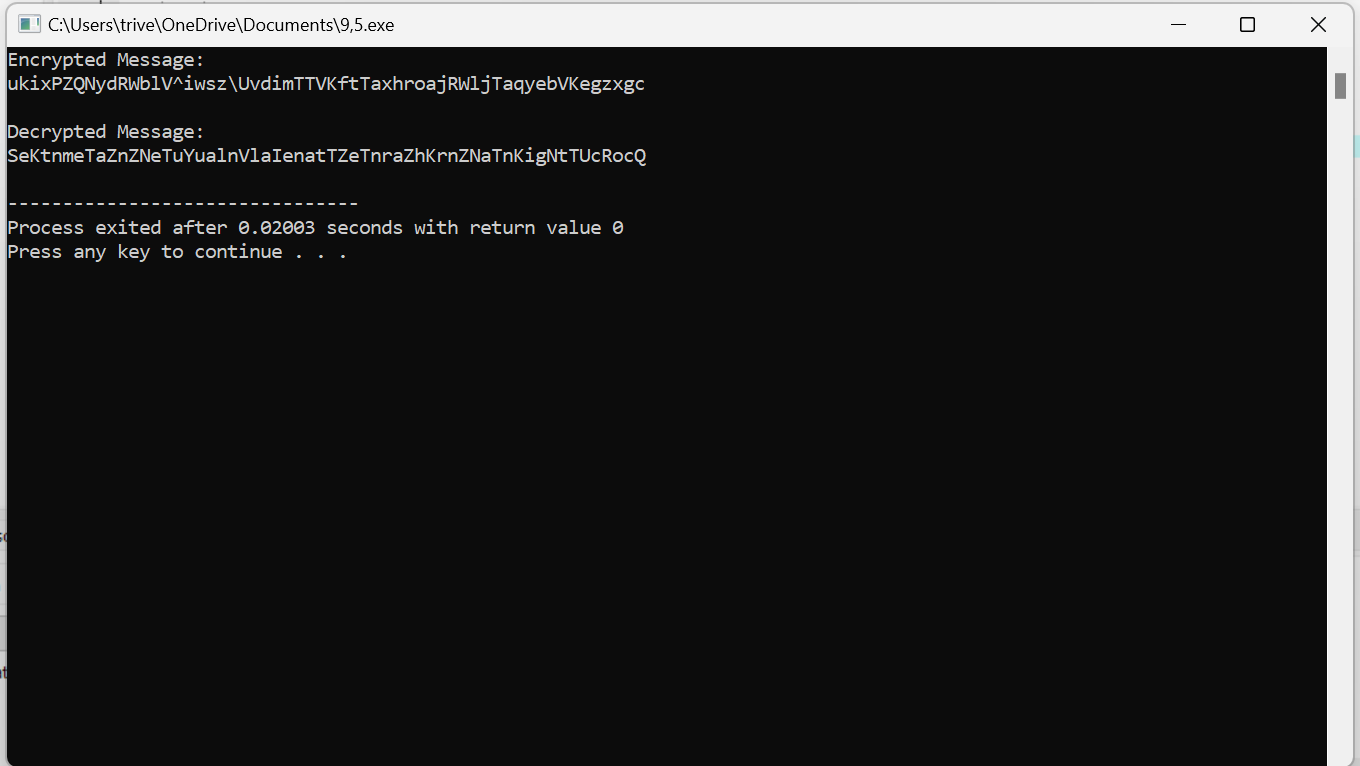
printf("Encrypted Message:\n%s\n\n", message);

decrypt(message, key);

printf("Decrypted Message:\n%s\n", message);

return 0;

}

Output:

3.

Write a High level code for Playfair matrix:

M F H I/J K

U N O P Q

Z V W X Y

E L A R G

D S T B C

Encrypt this message: Must see you over Cadogan West. Coming at once.

Programme:

#include <stdio.h>

#include <string.h>

void createPlayfairMatrix(char matrix[5][5]) {

char key[] = "MFH IJKUNOPQZVWXYELARGBCD";

int k = 0;

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

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

matrix[i][j] = key[k++];

}

}

}

void preprocessMessage(const char\* input, char\* message) {

int len = strlen(input);

int j = 0;

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

char ch = input[i];

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

message[j++] = ch;

}

}

message[j] = '\0';

}

void encryptPlayfair(char matrix[5][5], const char\* message, char\* ciphertext) {

int len = strlen(message);

int p = 0;

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

char firstLetter = message[i];

char secondLetter = (i + 1 < len) ? message[i + 1] : 'X';

int row1, col1, row2, col2;

for (int row = 0; row < 5; row++) {

for (int col = 0; col < 5; col++) {

if (matrix[row][col] == firstLetter) {

row1 = row;

col1 = col;

}

if (matrix[row][col] == secondLetter) {

row2 = row;

col2 = col;

}

}

}

if (row1 == row2) {

ciphertext[p++] = matrix[row1][(col1 + 1) % 5];

ciphertext[p++] = matrix[row2][(col2 + 1) % 5];

} else if (col1 == col2) {

ciphertext[p++] = matrix[(row1 + 1) % 5][col1];

ciphertext[p++] = matrix[(row2 + 1) % 5][col2];

} else {

ciphertext[p++] = matrix[row1][col2];

ciphertext[p++] = matrix[row2][col1];

}

}

ciphertext[p] = '\0';

}

int main() {

char matrix[5][5];

createPlayfairMatrix(matrix);

const char\* message = "Must see you over Cadogan West. Coming at once.";

char cleanedMessage[500];

char ciphertext[500];

preprocessMessage(message, cleanedMessage);

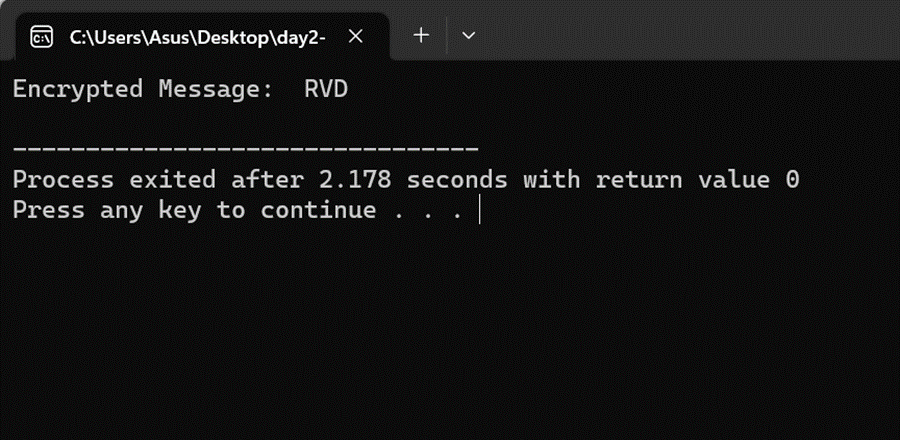
encryptPlayfair(matrix, cleanedMessage, ciphertext);

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

return 0;

}

Output:



6.

Write a C program for Hill cipher succumbs to a known plaintext attack if sufficient plaintext–

ciphertext pairs are provided. It is even easier to solve the Hill cipher if a chosen plaintext attack can be mounted.

Programme:

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void hillCipherEncrypt(char \*plainText, char \*keyMatrix) {

int i, j, k, len = strlen(plainText);

int key[2][2], plain[2], cipher[2];

for (i = 0, k = 0; i < 2; i++) {

for (j = 0; j < 2; j++, k++) {

key[i][j] = keyMatrix[k] - 'A';

}

}

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

plain[0] = plainText[i] - 'A';

plain[1] = plainText[i + 1] - 'A';

cipher[0] = key[0][0] \* plain[0] + key[0][1] \* plain[1];

cipher[1] = key[1][0] \* plain[0] + key[1][1] \* plain[1];

cipher[0] %= 26;

cipher[1] %= 26;

printf("%c%c", cipher[0] + 'A', cipher[1] + 'A');

}

}

int main() {

char plainText[100], keyMatrix[5];

printf("Enter plaintext (uppercase alphabets only): ");

scanf("%s", plainText);

printf("Enter 2x2 key matrix (uppercase alphabets only): ");

scanf("%s", keyMatrix);

if (strlen(plainText) % 2 != 0 || strlen(keyMatrix) != 4) {

printf("Plaintext and key matrix lengths must be even and 4 characters, respectively.\n");

return 1;

}

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

if (!isupper(plainText[i])) {

printf("Invalid characters in plaintext. Use uppercase alphabets only.\n");

return 1;

}

}

for (int i = 0; i < 4; i++) {

if (!isupper(keyMatrix[i])) {

printf("Invalid characters in the key matrix. Use uppercase alphabets only.\n");

return 1;

}

}

printf("Ciphertext: ");

hillCipherEncrypt(plainText, keyMatrix);

printf("\n");

return 0;

}

Output:

