EXPERIMENT-5:- Write a C program for generalization of the Caesar cipher, known as the affine Caesar cipher, has the following form: For each plaintext letter p, substitute the ciphertext letter C: = E([a, b], p) = (ap + b) mod 26 A basic requirement of any encryption algorithm is that it be one-to-one. That is, if p q, then E(k, p) E(k, q). Otherwise, decryption is impossible, because more than one plaintext character maps into the same ciphertext character. The affine Caesar cipher is not one-to-one for all values of a. For example, for a = 2 and b = 3, then E([a, b], 0) = E([a, b], 13) = 3.

a. Are there any limitations on the value of b?

b. Determine which values of a are not allowed.

Program:-

#include <stdio.h>

#include <string.h>

void railFenceEncrypt(const char \*message, int rails, char \*encrypted) {

int len = strlen(message);

int k = 0;

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

for (int j = i; j < len; j += rails) {

encrypted[k++] = message[j];

}

}

encrypted[len] = '\0';

}

void railFenceDecrypt(const char \*encrypted, int rails, char \*decrypted) {

int len = strlen(encrypted);

int k = 0;

int interval = (rails - 1) \* 2;

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

int step = interval - 2 \* i;

for (int j = i; j < len; j += interval) {

decrypted[j] = encrypted[k++];

if (step && step < interval && j + step < len) {

decrypted[j + step] = encrypted[k++];

}

}

}

decrypted[len] = '\0';

}

int main() {

char message[] = "HELLOWORLD";

int rails = 3;

char encrypted[100];

char decrypted[100];

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

railFenceEncrypt(message, rails, encrypted);

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

railFenceDecrypt(encrypted, rails, decrypted);

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

return 0;

}

Output:

