

RAIL-FENCE**Problem Statement**

The **rail fence cipher** (also called a **zigzag cipher**) is a form of transposition cipher. It derives its name from the way in which it is encoded. In the rail fence cipher, the plain text is written downwards and diagonally on successive “rails” of an imaginary fence, then moving up when the bottom rail is reached. When the top rail is reached, the message is written downwards again until the whole plaintext is written out. The message is then read off in rows. For example, if 3 “rails” and the message “HELLOWORLD” is used, the cipherer writes out:

H . . . O . . . L . .

E . L . W . R . D

. . L . . . O . . .

Then reads off to get the ciphertext:

HOLELWRDLO

Implement a program to perform this cipher.

Aim:

To implement Rail-Fence Cipher technique using C.

Algorithm:

1. Get the plaintext string from the user.
2. Take the string length of the plaintext.
3. For each plaintext character do the following-
 - a. If $ch \% 2 == 0$ put in a[] array
 - b. Else put in b[] array
4. Take each character in a[] array and put in s[] array and increment the index.
5. After all characters in a[] array are copied, then copy each character from b[] array and put into s[] array and increment the index.
6. Print the contents of s[] array to get ciphertext.

Program Code:

```
#include<stdio.h>
```

```
#include<string.h>
```

```
void encryptMsg(char msg[], int key){
```

```
    int msgLen = strlen(msg), i, j, k = -1, row = 0, col = 0;
```

```
    char railMatrix[key][msgLen];
```

```
    for(i = 0; i < key; ++i)
```

```
        for(j = 0; j < msgLen; ++j)
```

```

        railMatrix[i][j] = '\n';
    for(i = 0; i < msgLen; ++i){
        railMatrix[row][col++] = msg[i];

        if(row == 0 || row == key-1)

            k= k * (-1);

        row = row + k;
    }

    printf("\nEncrypted Message: ");

    for(i = 0; i < key; ++i)

        for(j = 0; j < msgLen; ++j)

            if(railMatrix[i][j] != '\n')

                printf("%c", railMatrix[i][j]);

    }

void decryptMsg(char enMsg[], int key){

    int msgLen = strlen(enMsg), i, j, k = -1, row = 0, col = 0, m = 0;

    char railMatrix[key][msgLen];

    for(i = 0; i < key; ++i)

        for(j = 0; j < msgLen; ++j)

            railMatrix[i][j] = '\n';

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

        railMatrix[row][col++] = '*';

        if(row == 0 || row == key-1)

            k= k * (-1);

        row = row + k;

```

```

    }

    for(i = 0; i < key; ++i)

        for(j = 0; j < msgLen; ++j)

            if(railMatrix[i][j] == '*')

                railMatrix[i][j] = enMsg[m++];

    row = col = 0;

    k = -1;

    printf("\nDecrypted Message: ");

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

        printf("%c", railMatrix[row][col++]);

        if(row == 0 || row == key-1)

            k = k * (-1);

        row = row + k;

    }

}

int main(){

    char msg[50];
    printf("plain text to encrypt:");
    scanf("%s",msg);

    int key = 3;

    printf("Original Message: %s", msg);

    encryptMsg(msg, key);

    char enMsg[50];
    printf("\ncipher text to decrypt:");
    scanf("%s",enMsg);

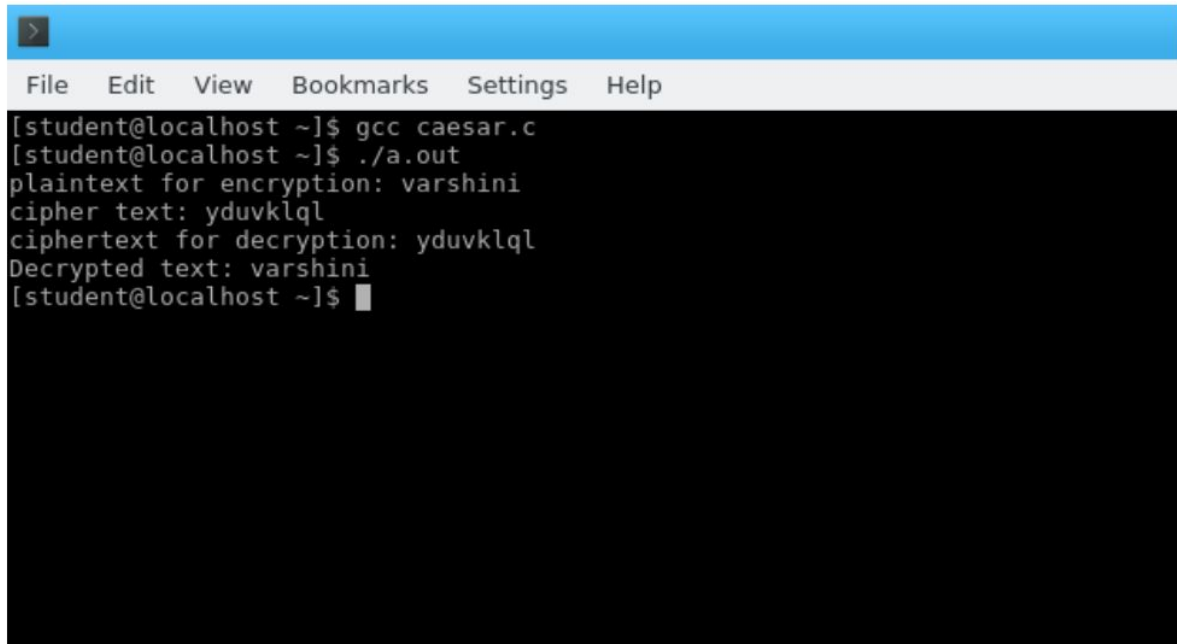
```

```
decryptMsg(enMsg, key);
```

```
return 0;
```

```
}
```

Output:

A screenshot of a terminal window with a blue title bar and a menu bar containing 'File', 'Edit', 'View', 'Bookmarks', 'Settings', and 'Help'. The terminal shows the following commands and output:

```
[student@localhost ~]$ gcc caesar.c
[student@localhost ~]$ ./a.out
plaintext for encryption: varshini
cipher text: yduvklql
ciphertext for decryption: yduvklql
Decrypted text: varshini
[student@localhost ~]$
```

Result: