Reg. no: 210701092

Experiment no: 1c Date: 6/2/24	Rail-fence Cipher

AIM:
To Write the C program to perform Rail fence Cipher for Both Encryption And Decryption Process

ALGORITHM:

Step 1: START

Step 2: Get the Message from the User

Step 3: Define two function Encryption And Decryption

Step 4: Arrage the plain text In the Zig Zag pattern

Step 5: Derving the Cipher Text by Adding the first Row of Arrangment with the Second Row of the Arrangment

Step 6: Get the Original text by Using the Cipher Text And Arrange in Zig Zag patter

Step 7: print the Output

Step 8: STOP

Reg no: 210701092   Name: Jeffrey Jesudasan R

Exp 1c : Rail-fence Cipher

Code:

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

// Function to encrypt a message using Rail Fence cipher
char* encryptRailFence(char* text, int key) {
    int textLength = strlen(text);
    char rail[key][textLength];
    char* result = (char*)malloc(textLength * sizeof(char));

    // Initialize rail matrix
    for (int i = 0; i < key; i++) {
        for (int j = 0; j < textLength; j++) {
rail[i][j] = '\n';
        }
    }

    // Fill the rail matrix
    bool dirDown = false;
    int row = 0, col = 0;
    for (int i = 0; i < textLength; i++) {
if (row == 0 || row == key - 1) {
dirDown = !dirDown;
        }
        rail[row][col++] = text[i];
        if (dirDown) {
            row++;
        } else {
            row--;
        }
    }

    // Construct the result
    int k = 0;
    for (int i = 0; i < key; i++) {
        for (int j = 0; j < textLength; j++) {
if (rail[i][j] != '\n') {
                result[k++] = rail[i][j];
            }
        }
    }
    result[k] = '\0';

    return result;
}

// Function to decrypt a Rail Fence cipher message
char* decryptRailFence(char* cipher, int key) {
    int cipherLength = strlen(cipher);
    char rail[key][cipherLength];
    char* result = (char*)malloc(cipherLength * sizeof(char));

    // Initialize rail matrix
    for (int i = 0; i < key; i++) {
        for (int j = 0; j < cipherLength; j++) {
rail[i][j] = '\n';
        }
    }

    // Mark the rail matrix
    bool dirDown = true;
    int row = 0, col = 0;
    for (int i = 0; i < cipherLength; i++) {
if (row == 0) {
            dirDown = true;
        }
        if (row == key - 1) {
            dirDown = false;
        }
        rail[row][col++] = '*';
        if (dirDown) {
            row++;
        } else {
            row--;
        }
    }

    // Fill the rail matrix with cipher text
    int index = 0;
    for (int i = 0; i < key; i++) {
        for (int j = 0; j < cipherLength; j++) {
            if (rail[i][j] == '*' && index < cipherLength) {
rail[i][j] = cipher[index++];
            }
        }
    }

    // Construct the result
    int k = 0;
    row = 0;
    col = 0;
    for (int i = 0; i < cipherLength; i++) {
if (row == 0) {
            dirDown = true;
        }
        if (row == key - 1) {
            dirDown = false;
        }
        if (rail[row][col] != '*') {
            result[k++] = rail[row][col++];
}
        if (dirDown) {
            row++;
        } else {
            row--;
}
    }
    result[k] = '\0';

    return result;
}

// Driver function
```

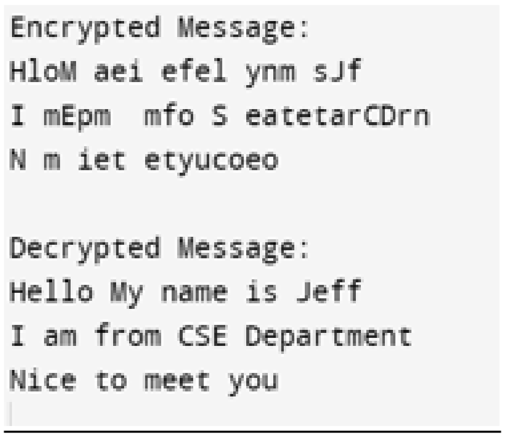
Reg no: 210701092   Name: Jeffrey Jesudasan R

```
int main() {
    // Encryption
    printf("Encrypted Message: \n");
    printf("%s\n", encryptRailFence("Hello My name is Jeff", 2));
    printf("%s\n", encryptRailFence("I am from CSE Department ", 3));
printf("%s\n", encryptRailFence("Nice to meet you", 3));

    // Decryption
    printf("\nDecrypted Message: \n");
    printf("%s\n", decryptRailFence("HloM aei efel ynm sJf", 2));
    printf("%s\n", decryptRailFence("I mEpm  mfo S eatetarCDrn", 3));
printf("%s\n", decryptRailFence("N m iet etyucoeo", 3));

    return 0;
}
```

Output:



```
Encrypted Message:
HloM aei efel ynm sJf
I mEpm  mfo S eatetarCDrn
N m iet etyucoeo

Decrypted Message:
Hello My name is Jeff
I am from CSE Department
Nice to meet you
```