**EXPERIMENT NO-8**

**OBJECTIVE**- a program to implement Rail Fence Cipher to encrypt & decrypt the given message by using the following conditions:

•Plaintext should be in lowercase.

•Ciphertext should be uppercase.

**SOURCE CODE-**

**def encryptRailFence(text, key):**

**# create the matrix to cipher**

**# plain text key = rows ,**

**# length(text) = columns**

**# filling the rail matrix**

**# to distinguish filled**

**# spaces from blank ones**

**rail = [['\n' for i in range(len(text))]**

**for j in range(key)]**

**# to find the direction**

**dir\_down = False**

**row, col = 0, 0**

**for i in range(len(text)):**

**# check the direction of flow**

**# reverse the direction if we've just**

**# filled the top or bottom rail**

**if (row == 0) or (row == key - 1):**

**dir\_down = not dir\_down**

**# fill the corresponding alphabet**

**rail[row][col] = text[i]**

**col += 1**

**# find the next row using**

**# direction flag**

**if dir\_down:**

**row += 1**

**else:**

**row -= 1**

**# now we can construct the cipher**

**# using the rail matrix**

**result = []**

**for i in range(key):**

**for j in range(len(text)):**

**if rail[i][j] != '\n':**

**result.append(rail[i][j])**

**return("" . join(result))**

**# This function receives cipher-text**

**# and key and returns the original**

**# text after decryption**

**def decryptRailFence(cipher, key):**

**# create the matrix to cipher**

**# plain text key = rows ,**

**# length(text) = columns**

**# filling the rail matrix to**

**# distinguish filled spaces**

**# from blank ones**

**rail = [['\n' for i in range(len(cipher))]**

**for j in range(key)]**

**# to find the direction**

**dir\_down = None**

**row, col = 0, 0**

**# mark the places with '\*'**

**for i in range(len(cipher)):**

**if row == 0:**

**dir\_down = True**

**if row == key - 1:**

**dir\_down = False**

**# place the marker**

**rail[row][col] = '\*'**

**col += 1**

**# find the next row**

**# using direction flag**

**if dir\_down:**

**row += 1**

**else:**

**row -= 1**

**# now we can construct the**

**# fill the rail matrix**

**index = 0**

**for i in range(key):**

**for j in range(len(cipher)):**

**if ((rail[i][j] == '\*') and**

**(index < len(cipher))):**

**rail[i][j] = cipher[index]**

**index += 1**

**# now read the matrix in**

**# zig-zag manner to construct**

**# the resultant text**

**result = []**

**row, col = 0, 0**

**for i in range(len(cipher)):**

**# check the direction of flow**

**if row == 0:**

**dir\_down = True**

**if row == key-1:**

**dir\_down = False**

**# place the marker**

**if (rail[row][col] != '\*'):**

**result.append(rail[row][col])**

**col += 1**

**# find the next row using**

**# direction flag**

**if dir\_down:**

**row += 1**

**else:**

**row -= 1**

**return("".join(result))**

**# Driver code**

**if \_\_name\_\_ == "\_\_main\_\_":**

**print(encryptRailFence("attack at once", 2))**

**print(encryptRailFence("GeeksforGeeks ", 3))**

**print(encryptRailFence("defend the east wall", 3))**

**# Now decryption of the**

**# same cipher-text**

**print(decryptRailFence("GsGsekfrek eoe", 3))**

**print(decryptRailFence("atc toctaka ne", 2))**

**print(decryptRailFence("dnhaweedtees alf tl", 3))**

**OUTPUT:**

atc toctaka ne

GsGsekfrek eoe

dnhaweedtees alf tl

**Developed by:** Abhishek Pandey