Railfence

```
# function to encrypt a message
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("Hello World", 2))
```