---------------------------------------------------------------------------

**Name : Atharva Paliwal**

**Roll No : 40 [5B]**

---------------------------------------------------------------------------

**\*\*\* EXPERIMENT NO: 02 \*\*\***

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**AIM- Write a Programme to implement Bit Stuffing and Charachter Stuffing.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CODE-**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CHARACTER STUFFING**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def char\_stuff(msg,flg): ## Charachter Stuffing

stf\_msg=flg #initialising first char as flag

for i in range(len(msg)):

if msg[i]==flg:

stf\_msg=stf\_msg+flg

stf\_msg=stf\_msg+msg[i]

stf\_msg=stf\_msg+flg

return(stf\_msg) ##return the msg with stuffed char

def char\_destuff(stf\_msg,flg): ## Charachter De-stuffing

destf\_msg=''

key=0

stf\_msg=stf\_msg[1:len(stf\_msg)-1]

for i in stf\_msg:

if i==flg and key==0:

key=1

continue

if i==flg:

key=0

destf\_msg=destf\_msg+i

return(destf\_msg) ##return the msg with destuffed char

#Driver Code

msg=input('Enter Your Message : ')

flg=input('Enter the flag Character : ')

stf\_msg=char\_stuff(msg,flg)

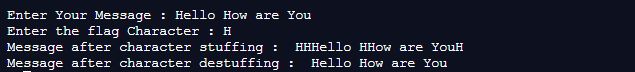
print('Message after character stuffing : ',stf\_msg)

destf\_msg=char\_destuff(stf\_msg,flg)

print('Message after character destuffing : ',destf\_msg)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**OUTPUT-**



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**BIT STUFFING**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def createBitString(message): #to create bit sequence from a string

bit\_string = ''

for c in message:

#to balance the sequence of 7 , '0' bit is added if ascii value < 64

if ord(c) < 64:

bit\_string += '0'

#format() returns 6-bit binary sequence if ascii value is < 64.

bit\_string += ''.join(format(ord(c),'b'))

return bit\_string

def createAsciiString(bit\_string): #to create string from a bit sequence

result = ''

for i in range(0,len(bit\_string),7):

c = chr(int(bit\_string[i:i+7], 2))

result += c

#print(result)

return result

def stuffBit(message): #to stuff bit as an esc bit where ever needed

msg = list(message)

count = 0

i = 0

while i != len(msg):

if msg[i] == '1':

count += 1

else :

count = 0

if count == 5:

msg.insert(i+1,'0')

count = 0

i += 1

return (''.join(msg))

def destuffBit(message): #to destuff the stuffed bits

msg = list(message)

count = 0

i = 0

while i != len(msg):

if msg[i] == '1':

count += 1

else : count = 0

if count == 5:

msg.pop(i+1)

count = 0

i += 1

return (''.join(msg))

#Driver Code

message = input('Input Data to send: ')

bit\_string = createBitString(message)

print('Bit String: '+bit\_string+'\n')

stuffed\_bit\_msg = stuffBit(bit\_string)

print('Bit String after Bit Stuffing: '+stuffed\_bit\_msg+'\n')

print('Data after Bit Stuffing:',createAsciiString(stuffed\_bit\_msg))

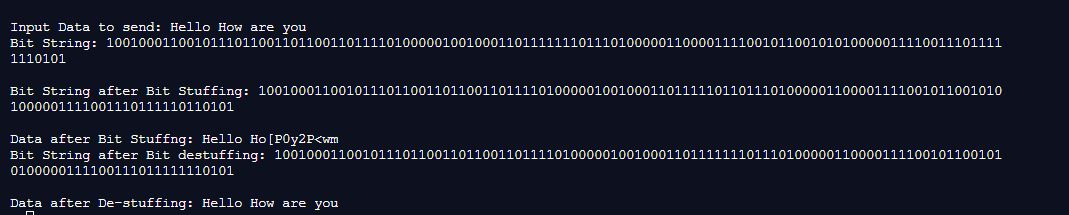
destuffed\_bit\_msg = destuffBit(stuffed\_bit\_msg)

print('Bit String after Bit destuffing: '+destuffed\_bit\_msg+'\n')

print('Data after De-stuffing:',createAsciiString(destuffed\_bit\_msg))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**OUTPUT-**



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*