
Name   : Atharva Paliwal

Roll No   : 40 [5B]

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

****

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

****

CODE-

****

CHARACTER STUFFING

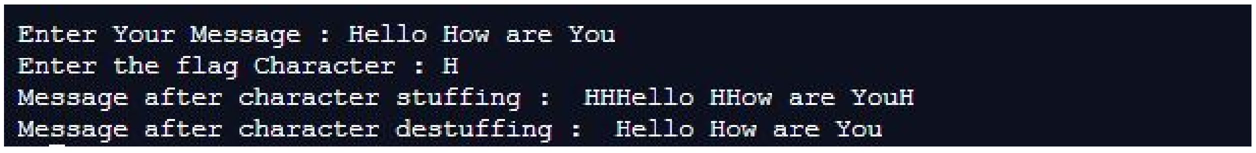
****

```
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-



```
Enter Your Message : Hello How are You
Enter the flag Character : H
Message after character stuffing :  HHHello HHow are YouH
Message after character destuffing :  Hello How are You
```

****

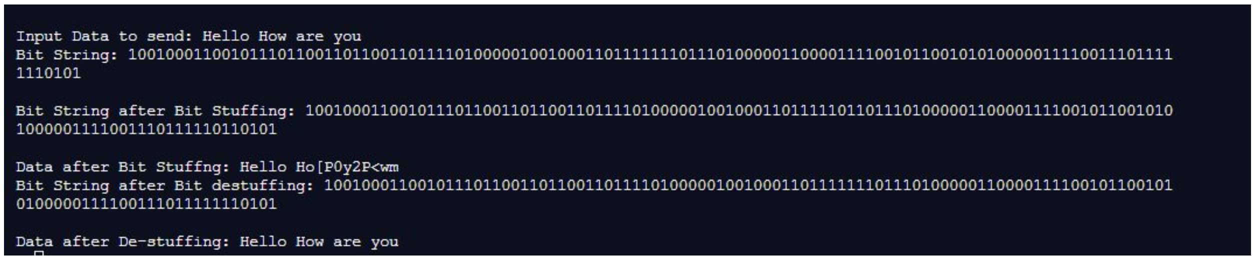
```
***************************************************************************

                              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-



```
Input Data to send: Hello How are you
Bit String: 100100011001011110110011011001101111010000010010001101111111011110100000110000111100101100101010000011110011101111
1110101

Bit String after Bit Stuffing: 10010001100101110110011011001101111010000010010001101111101101110100000110000111100101100101O
10000011110011101111110110101

Data after Bit Stuffng: Hello Ho[P0y2P<wm
Bit String after Bit destuffing: 100100011001011101100110110011011110100000100100011011111110111010000011000011110010110010l
010000011110011101111111110101

Data after De-stuffing: Hello How are you
```

****