

Name: SHASHWAT NIMESH SHAH SAPID: 60004220126

**Experiment No: 03 Aim:** Write a program to implement Framing Techniques:

Character count, Byte stuffing, Bit stuffing

# Explain character count with example. Theory:

This method is rarely used and is generally required to count total number of characters that are present in

frame. This is be done by using field in header. Character count method ensures data link layer at the receiver or destination about total number of characters that follow, and about where the frame ends. There is disadvantage also of using this method i.e., if anyhow character count is disturbed or distorted by an error occurring during transmission, then destination or receiver might lose synchronization. The destination or receiver might also be not able to locate or identify beginning of next frame.

# Code:

import java.util.\*; class Main {

public static void main(String[] args) { Scanner sc=new Scanner(System.in);

System.out.print("Enter the number of strings: "); int n=sc.nextInt();

String s[]=new String[n]; int a[]=new int[n];

System.out.println("Enter strings:"); for(int i=0;i<n;i++)

{

s[i]=sc.next();

a[i]=s[i].length();

}

System.out.println("Output:"); for(int i=0;i<n;i++)

{



System.out.print((a[i]+1)+s[i]);

}

}

}

# Output:

**Explain Byte stuffing with example. Theory:**

Character stuffing is also known as byte stuffing or character-oriented framing and is same as that of bit

stuffing but byte stuffing actually operates on bytes whereas bit stuffing operates on bits. In byte stuffing, special byte that is basically known as ESC (Escape Character) that has predefined pattern is generally added to data section of the data stream or frame when there is message or character that has same pattern as that of flag byte.

But receiver removes this ESC and keeps data part that causes some problems or issues. In simple words, we can say that character stuffing is addition of 1 additional byte if there is presence of ESC or flag in text.

# Code:

#include <stdio.h> #include <stdlib.h> #include <string.h>

int main()

{

char a[30],fs[50]="",t[3],sd,ed,x[3],s[3],d[3],y[3]; int i;

printf("Enter characters to be stuffed: "); scanf("%s",a);

printf("Enter starting delimeter: ");



scanf(" %c",&sd);

printf("Enter ending delimeter: "); scanf(" %c",&ed);

x[0]=s[0]=s[1]=sd;

x[1]=s[2]='\0';

y[0]=d[0]=d[1]=ed;

d[2]=y[1]='\0';

strcat(fs,x);

for(i=0;i<strlen(a);i++){ t[0]=a[i];

t[1]='\0';

if(t[0]==sd){

strcat(fs,s);

}else if(t[0]==ed){ strcat(fs,d);

}else{ strcat(fs,t);

}

}

strcat(fs,y);

printf("\nAfter Stuffing\n %s",fs); return 0;

}

# Output:



**Explain Bit stuffing with example. Theory:**

Bit stuffing is also known as bit-oriented framing or bit-oriented approach. In bit stuffing, extra bits are

being added by network protocol designers to data streams. It is generally insertion or addition of extra bits into transmission unit or message to be transmitted as simple way to provide and give signaling information and data to receiver and to avoid or ignore appearance of unintended or unnecessary control sequences.

It is type of protocol management simply performed to break up bit pattern that results in transmission to go out of synchronization. Bit stuffing is very essential part of transmission process in network and communication protocol. It is also required in USB.

# Code:

import java.io.\*; import java.util.\*;

class Main {

public static void main (String[] args) { int n, j=0, k;

Scanner sc = new Scanner(System.in); System.out.print("Enter the length of string: "); n = sc.nextInt();

int a[] = new int[n]; int b[] = new int[100];

System.out.println("Enter string: "); for(int i=0; i<n; i++) {

a[i] = sc.nextInt();

}

for(int i=0; i<n; i++) { if(a[i]==1) {

int count=1; b[j] = a[i]; j++;

for(k=i+1; k<n && count<5; k++) { if(a[k] == 1) {

b[j] = a[k]; count++; j++;



}

}

if(count == 5) { j++;

b[j+1] = 0;

}

i=k;

}

else {

b[j] = a[i]; j++;

}

}

System.out.println("After bit stuffing: "); for(int i=0; i<j; i++) {

System.out.print(b[i]);

}

}

}

# Output:

**Conclusion:**

Different framing methods are implemented.