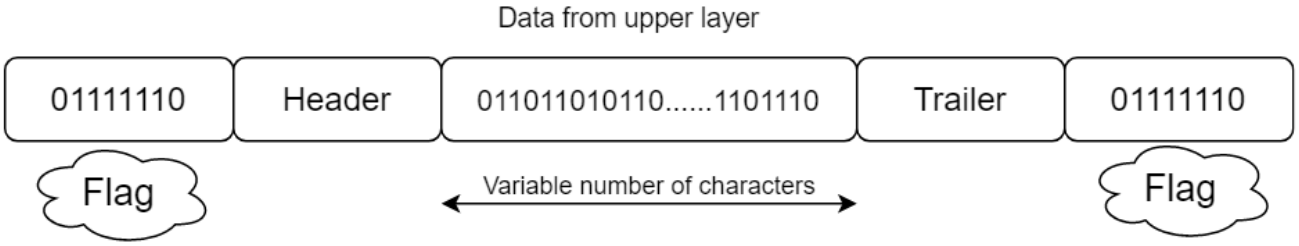
Bit Stuffing

Whenever the sender's Data link layer encounters five consecutive 1's in the data, it automatically stuffs a 0 bit into the outgoing bitstream which is called payloads. If 6 1's come then at the receiving end it might get misinterpreted that it is an end flag, So after the occurrence of each consecutive 5 1's one zero will be stuffed & that's why it is called Bit stuffing.




```
import java.util.*;
public class BitStuffing {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the data stream: ");
        String dataStream = sc.nextLine();
        String stuffedStream = bitStuffing(dataStream);
        System.out.println("Original Data Stream: " + dataStream);
        System.out.println("Stuffed Data Stream: " + stuffedStream);
        sc.close();    }
    public static String bitStuffing(String dataStream) {
        StringBuilder stuffedStream = new StringBuilder();
        int consecutiveOnes = 0;
        for (int i = 0; i < dataStream.length(); i++) {
            char bit = dataStream.charAt(i);
            stuffedStream.append(bit);
            consecutiveOnes = (bit == '1') ? consecutiveOnes + 1 : 0;
            if (consecutiveOnes == 5) {
                stuffedStream.append('0');
                consecutiveOnes = 0;
            }       }
        return stuffedStream.toString();
    }   }
```

Character Stuffing

Character stuffing is a technique used in computer networks to ensure that data frames are correctly interpreted by the receiving end. It involves adding special control characters to the data to distinguish them from control characters used for signaling or framing.

Here's a simple explanation with an example:

Imagine you're sending a message "HELLO" over a network. The receiver needs to know where the message starts and ends. To distinguish the actual data from control characters, you can use character stuffing.

1. **Original Message:** HELLO
2. **Stuffed Message:** DLE HELLO DLE

In this example, DLE (Data Link Escape) is a control character used for stuffing. It indicates that the following character is part of the data, not a control character. So, the receiver knows that the actual message is "HELLO" and not a control character followed by "ELLO".

Character stuffing ensures that data frames are correctly interpreted even if they contain control characters, improving the reliability of communication over the network.
DLE (Data Link Escape), STX (Start of Text), and ETX (End of Text) are control characters used in character-oriented protocols, such as the ASCII protocol, to control the flow of data and to mark the beginning and end of a message.

1. **DLE (Data Link Escape):** It is used to indicate that the following character should be interpreted differently. It is often used in character stuffing to distinguish data from control characters.
2. **STX (Start of Text):** It marks the beginning of a text or message. It indicates the start of the actual data to be transmitted.
3. **ETX (End of Text):** It marks the end of a text or message. It indicates that the transmission of the data is complete.

These characters are used to frame messages and ensure that the receiver can correctly interpret the data being transmitted. For example, in a protocol that uses DLE for character stuffing, STX might indicate the start of a message, followed by the actual data, and ETX would mark the end of the message.

```
import java.util.*;

class Char

{

   public static void main(String r[])

   {

      Scanner sc=new Scanner(System.in);

      System.out.println("Enter number of characters: ");

      int n=sc.nextInt();

      String in[]=new String[n];

      System.out.println("Enter characters: ");

      for(int i=0;i<n;i++)

      {
```

```
            in[i]=sc.next();
        }
        for(int i= 0;i<n;i++)
        {
            if(in[i].equals("dle"))
            {
                in[i]="dle dle";
            }       }
        System.out.println("Transmitted message is: ");
        System.out.print(" dle stx ");
        for(int i=0;i<n;i++)
        {
            System.out.print(in[i]+" ");
        }
        System.out.println(" dle etx ");
    }
}
```