Name: Hitendra Sisodia
Sap id: 500091910

Lab 2

LAB – 2(Study and implementation of Bit Stuffing and De-stuffing)

Bit stuffing is the insertion of non-information bits into data. Note that stuffed bits should not be confused with overhead bits.

Applications of Bit Stuffing –

1. synchronize several channels before multiplexing
2. rate-match two single channels to each other
3. run length limited coding

Example of bit stuffing –
Bit sequence: 11010111110101111101011111110 (without bit stuffing)
Bit sequence: 11010111110010111111010111111**0**110 (with bit stuffing)

Implementation of Bit Stuffing in c:

```
#include<iostream>
using namespace std;
int main()
{
    int bits;
    cout<<"Enter the Total number of bits: ";
    cin>>bits;
    int arr[100];
    bool val = true;
    for(int i = 0 ; i < bits ; i++){
        cout<<"Enter the " <<i+1<<" ith bits: ";
        int input = 0;
        cin>>input;
        if(input == 1 || input == 0){
            arr[i] = input;
        }
        else{
            cout<<"Please enter valid bits";
            val = false;
            break;
        }
    }
    int count = 0;
    for(int i = 0 ; i < bits; i++){
        if(arr[i] == 1){
            count++;
        }
        else if(arr[i] == 0){
            count = 0;
        }
        if(count == 5 && arr[i+1] == 1 || arr[]){
            for(int j = bits ; j > i ; j--){
                arr[j+1] = arr[j];
            }
            arr[i] = 0;
            bits++;
            count = 0;
        }
    }
```

Name: Hitendra Sisodia
Sap id: 500091910

Lab 2

```
    // Loop for display
    if(val){
        for(int i = 0 ; i < bits ; i++){
            cout<<arr[i];
    }
    }
}
```

Output:

```
PS C:\Users\himan> cd "c:\Users\himan\OneDrive - UPES\Desktop\UPES 4th_Sem(2)\DCN Lab\Lab 2\" ; if ($?)
 ($?) { .\Lab-2 }
Enter the Total number of bits: 12
Enter the 1 ith bits: 1
Enter the 2 ith bits: 1
Enter the 3 ith bits: 1
Enter the 4 ith bits: 1
Enter the 5 ith bits: 1
Enter the 6 ith bits: 1
Enter the 7 ith bits: 1
Enter the 8 ith bits: 1
Enter the 9 ith bits: 1
Enter the 10 ith bits: 1
Enter the 11 ith bits: 1
Enter the 12 ith bits: 1
11110111101111
```

Name: Hitendra Sisodia
Sap id: 500091910

Lab 2

Bit Destuffing or **Bit Unstuffing** is a process of undoing the changes in the array made during the bit stuffing process i.e., removing the extra **0** bit after encountering **5** consecutive **1's**.

ARR = [0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1]
Output = [0, 1, 0, 1, 1, 1, 1, 1, 1, 1]
Explanation
We can see from index 3 we have five consecutive 1s and 2 consecutive 1s from index 9 which are seperated by a 0. Hence, we will remove zero at index 8.

ARR = [1, 1, 0, 1, 1, 1, 1, 1, 0]
Output = [1, 1, 0, 1, 1, 1, 1, 1]
Explanation
As there are no 1s seperated by five consecutive 1s starting from index 3. Hence Zero at position 8 will not be removed.

Implementation of Bit Destuffing in c:

```c
#include <stdio.h>
#include <string.h>
#include<iostream>
using namespace std;
void bitDestuffing(int N, int arr[]){
    int brr[30];
    int i, j, k;
    i = 0;
    j = 0;
    int count = 1;
    while (i < N) {
        if (arr[i] == 1) {
            brr[j] = arr[i];
            for (k = i + 1; arr[k] == 1 && k < N && count < 5; k++) {
                j++;
                brr[j] = arr[k];
                count++;
                if (count == 5) {
                    k++;
                }
                i = k;
            }
        }
        }
        i++;
        j++;
    }
    for (i = 0; i < j; i++)
        printf("%d", brr[i]);
}
int main(){
    int N = 7;
    int arr[] = { 1, 1, 1, 1, 1, 0, 1 };
    bitDestuffing(N, arr);
    return 0;
}
```

Name: Hitendra Sisodia
Sap id: 500091910

Lab 2

Output:

```
PS C:\Users\himan> cd "c:\Users\himan\OneDrive - UPES\Desktop\UPES 4th_Sem(2)\DCN Lab\Lab 2\" ; if ($?)
 ($?) { .\Lab-2 }
111111
PS C:\Users\himan\OneDrive - UPES\Desktop\UPES 4th_Sem(2)\DCN Lab\Lab 2>
```