Bit Stuffing

```c
#include <stdio.h>
#include <string.h>
int main(){
    char data[50], stuff[50], dstuff[50];
    int i, j, count, length;
    printf("Enter the data\n");
    scanf("%s", data);
    length = strlen(data); // length of the data
    count = 0, j = 0;
    for (i = 0; i < length; i++){
        if (data[i] == '1')
            count++;
        else
            count = 0;
        stuff[j] = data[i];
        j++;
        if (count == 5 && data[i + 1] == '1'){
            stuff[j] = '0';
            j++;
            count = 0;
        }
    }
    stuff[j] = '\0';
    printf("stuffed data at intermediate site is \n 01111110   %s 
01111110 \n", stuff);
}
```

Character Stuffing

```c
#include<stdio.h>
#include<string.h>
int main(){
    int i = 0, j = 0, n, pos;
    char a[20], b[50], ch;
    printf("enter string\n");
    scanf("%s", a);
    n = strlen(a);
    printf("enter position\n");
    scanf("%d", &pos);
    if (pos > n){
        printf("invalid position, Enter again :");
        scanf("%d", &pos);
    }
    printf("enter the character\n");
    ch = getchar();
    b[0] = 'd';
    b[1] = 'l';
    b[2] = 'e';
    b[3] = 's';
    b[4] = 't';
    b[5] = 'x';
    j = 6;
    while (i < n){
        if (i == pos - 1){
            b[j] = 'd';
            b[j + 1] = 'l';
            b[j + 2] = 'e';
            b[j + 3] = ch;
            b[j + 4] = 'd';
            b[j + 5] = 'l';
            b[j + 6] = 'e';
            j = j + 7;
        }
        if (a[i] == 'd' && a[i + 1] == 'l' && a[i + 2] == 'e'){
            b[j] = 'd';
            b[j + 1] = 'l';
            b[j + 2] = 'e';
            j = j + 3;
        }
        b[j] = a[i];
        i++;
        j++;
    }
    b[j] = 'd';
    b[j + 1] = 'l';
    b[j + 2] = 'e';
    b[j + 3] = 'e';
    b[j + 4] = 't';
    b[j + 5] = 'x';
    b[j + 6] = '\0';
    printf("\nframe after stuffing:\n");
    printf("%s\n", b);
}
```

Distance Vector

```
#include <stdio.h>
#include <stdlib.h>
struct node{
    int dist[15];
    int from[15];
};

int main()
```

```
{
    int a[15][15],n=0,i,j,k,count;
    struct node s[10];
    printf("enter number of nodes\n");
    scanf("%d",&n);
    printf("enter matrix\n");
    for(i=1;i<=n;i++){
        for(j=1;j<=n;j++){
            scanf("%d",&a[i][j]);
            s[i].dist[j]=a[i][j];//read it as from i to j distance is a[i]
[j]
            s[i].from[j]=j; // read it as from i to j next node is j
        }
    }
    do{
        count=0;
        for(k=1;k<=n;k++){
            for(i=1;i<=n;i++){
                for(j=1;j<=n;j++){
                    if(s[i].dist[j] > a[i][k] + s[k].dist[j]){
                        s[i].dist[j] = a[i][k] + s[k].dist[j];
                        s[i].from[j]=k;
                        count++;
                    }
                }
            }
        }
    }while(count!=0);
    for(i=1;i<=n;i++){
        for(j=1;j<=n;j++){
            printf("Src : %d -> Dest : %d Next node : %d Distance :
%d\n",i,j,s[i].from[j],s[i].dist[j]);
        }
        printf("\n");
    }
}
```

CRC-CCIT

```
// CRC : Cyclic Redundancy Check - data error detection
// CCITT : Consultative Committee for International Telephony and
Telegraphy
#include<stdio.h>
int a[100], b[100], i, j, len, k, count = 0;
//Generator Polynomial:g(x)=x^16+x^12+x^5+1
int gp[] = { 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1 };
int main() {
  void div();
  printf("\nEnter the length of Data Frame :");
  scanf("%d", &len);
  printf("\nEnter the Message :");
  for (i = 0; i < len; i++)
    scanf("%d", &a[i]);
  for (i = 0; i < 16; i++)
    a[len++] = 0;
  for (i = 0; i < len; i++)
    b[i] = a[i];
  k = len - 16;
  div();
  for (i = 0; i < len; i++)
    b[i] = b[i] ^ a[i];
  printf("\nData to be transmitted : ");
  for (i = 0; i < len; i++)
    printf("%2d", b[i]);
  printf("\n\nEnter the Reveived Data : ");
  for (i = 0; i < len; i++)
    scanf("%d", &a[i]);
  div();
  for (i = 0; i < len; i++)
    if (a[i] != 0){
      printf("\nERROR in Recived Data");
    return 0;
    }
  printf("\nData Recived is ERROR FREE");
}

void div(){
  for (i = 0; i < k; i++){
    if (a[i] == gp[0]){
      for (j = i; j < 17 + i; j++)
        a[j] = a[j] ^ gp[count++];
    }
    count = 0;
  }
}
```

Stop and Wait

```
#include<stdio.h>
#include<stdlib.h>
#include <unistd.h>

int main()
{
    int i, j, noframes, x, wait;
    printf("Enter the number of frames: ");
    scanf("%d", &noframes);
    i = 1;
    j = 1;
    printf("Number of frames is %d ", noframes);
    while (noframes > 0){
        printf("\nSending frame : %d", i);
        x = rand() % 10; // random number between 0 to 9
        if (x % 10 == 0){
            for (wait = 1;wait < 2;wait++){
                printf("\nWaiting for %d seconds\n", wait);
                sleep(wait); // sleep() function is used to pause the
program for a given number of seconds
            }
            printf("\nSending frame : %d\n", i);
            x = rand() % 10;
        }
        printf("\nAck for frame %d\n", j);
        noframes--;
        i++;
        j++;
    }
    printf("\nEnd of stop and wait protocol\n");
}
```

Sliding

```
#include<stdio.h>
int main(){
    int w, i, f, frames[50];
    printf("Enter window size: ");
    scanf("%d", &w);
    printf("\nEnter number of frames to transmit: ");
    scanf("%d", &f);
    printf("\nEnter %d frames: ", f);
    for (i = 1; i <= f; i++)
        scanf("%d", &frames[i]);
    printf("\nWith sliding window protocol the frames will be sent in the
following manner (assuming no corruption of frames)\n\n");
    printf("After sending %d frames at each stage sender waits for
acknowledgement sent by the receiver\n\n", w);
    for (i = 1; i <= f; i++){
        if (i % w == 0){
            printf("%d\n", frames[i]);
            printf("Acknowledgement of above frames sent is received by
sender\n\n");
        }
        else
            printf("%d ", frames[i]);
    }
    if (f % w != 0)
        printf("\nAcknowledgement of above frames sent is received by
sender\n");
    return 0;
}
```

Leaky Bucket

```c
#include<stdio.h>
#include<stdlib.h>
#define MIN(x,y) (x>y)?y:x
int main(){
    int orate, drop = 0, cap, x, count = 0, inp[10] = { 0 }, i = 0, nsec, ch;
    printf("\n enter bucket size : ");
    scanf("%d", &cap);
    printf("\n enter output rate :");
    scanf("%d", &orate);
    do{
        printf("\n enter number of packets coming at second %d :", i + 1);
        scanf("%d", &inp[i]);
        i++;
        printf("\n enter 1 to contiue or 0 to quit.........");
        scanf("%d", &ch);
    } while (ch);
    nsec = i;
    printf("\n second \t recieved \t sent \t dropped \t remained \n");
    for (i = 0;count || i < nsec;i++){
        printf("    %d", i + 1);
        printf(" \t%d\t ", inp[i]);
        printf(" \t %d\t ", MIN((inp[i] + count), orate));
        if ((x = inp[i] + count - orate) > 0){
            if (x > cap){
                count = cap;
                drop = x - cap;
            }
            else{
                count = x;
                drop = 0;
            }
        }
        else{
            drop = 0;
            count = 0;
        }
        printf(" \t %d      \t %d \n", drop, count);
    }
    return 0;
}
```