Week-1(1) Bit Stuffing

```
#include <stdio.h>
#include<string.h>
int main()
{
    int a[20],b[30],i,j,k,count,n;
    printf("Enter frame length: ");
    scanf("%d",&n);
    printf("Enter input frame(0's & 1's): ");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    i=0;
    count=1;
    j=0;
    while(i<n){
        if(a[i]==1){
            b[j] = a[i];
            for(k=i+1;a[k]==1 && k<n && count<5;k++){
                j++;
                b[j]=a[k];
                count++;
                if(count == 5){
                    j++;
                    b[j]=0;
                    count=0;
                }

                i=k;
            }
        }else{
            b[j]=a[i];
        }
        i++;
        j++;
    }
    printf("After stuffing the frame is: ");
    for(i=0;i<j;i++){
        printf("%d",b[i]);
    }

    return 0;
}
```

Week-1(2) Character-stuffing

Character stuffing through C

```
#include <stdio.h>
#include<string.h>
int main()
{
    char a[30],b[]="dle";
    int n,i,j,p,count=0;
    printf("Enter frame: ");
    scanf("%s",a);
    n=strlen(a);
    printf("Frame after stuffing : \n");
    printf("DLESTX ");
    for(i=0;i<n;i++){
        count=0;
        p=i;
        for(j=0;j<3;j++){
            if(a[i]==b[j]){
                count++;
                i++;
            }
        }
        if(count!=3){
            i=p;
        }
        if(count==3){
            printf("dledle");
            i--;
        }else{
            printf("%c",a[i]);
        }
    }
    printf(" DLEETX");



    return 0;
}
```

WEEK-2 CRC

```
//PROGRAM FOR CYCLIC REDUNDENCY CHECK
#include<stdio.h>

int main ()
{
   int n, m;
   printf ("Enter frame length and key length: ");
   scanf ("%d %d", &n, &m);

   int fr[n], ky[m], rem[m], dupFr[n + m - 1], rec[n + m - 1];
   int i, j, y, z, dupFrLen = (n + m - 1);
   printf ("Enter Frame: ");

   for (i = 0; i < n; i++)
   {
      scanf ("%d", &fr[i]);
   }
   printf ("Enter key: ");

   for (i = 0; i < m; i++)
   {
      scanf ("%d", &ky[i]);
   }
   for (i = 0; i < dupFrLen; i++)
   {
      if (i < n)
            {
         dupFr[i] = fr[i];

      }
      else
            {
         dupFr[i] = 0;

      }

   }

//division
   for (j = 0; j < n; j++)
   {

      if (dupFr[j] == 1)
         {

         for (y = 0, z = j; y < m; y++, z++)
               {
            rem[y] = dupFr[z] ^ ky[y];
            dupFr[z] = rem[y];
```

```
        }

      }

   }

   //print frame, key, dupFr
   printf ("----------------------\nFrame: ");

   for (i = 0; i < n; i++)
   {
      printf ("%d", fr[i]);
   }
   printf ("\nkey: ");
   for (i = 0; i < m; i++)
   {
      printf ("%d", ky[i]);
   }
   printf ("\nDupFr: ");

   for (i = 0; i < dupFrLen; i++)
   {
      printf ("%d", dupFr[i]);
   }


//reciver
   for (i = 0; i < dupFrLen; i++)
   {
      if (i < n)
         {
         rec[i] = fr[i];
      }
      else
         {
         rec[i] = dupFr[i];
      }
   }
   printf ("\nrecieved: ");

   for (i = 0; i < dupFrLen; i++)
   {
      printf ("%d", rec[i]);
   }

//division
   for (j = 0; j < n; j++)
   {
      if (rec[j] == 1)
```

```
            {
          for (y = 0, z = j; y < m; y++, z++)
                {
             rem[y] = rec[z] ^ ky[y];
             rec[z] = rem[y];
          }
       }
   }

//rec print
    printf ("\nrec rem : ");
    for (i = 0; i < dupFrLen; i++)
    {
       printf ("%d", rec[i]);
    }
    for (i = 0; i < dupFrLen; i++)
    {
       if (rec[i])
             {
          printf ("\nRecieved frame is wrong");
          break;
       }
    }
    printf ("\nRecieved frame is correct");
    return 0;
}



/*
INPUT:
9 5
1 1 0 1 0 1 1 1 1
1 0 0 1 1

*/
```

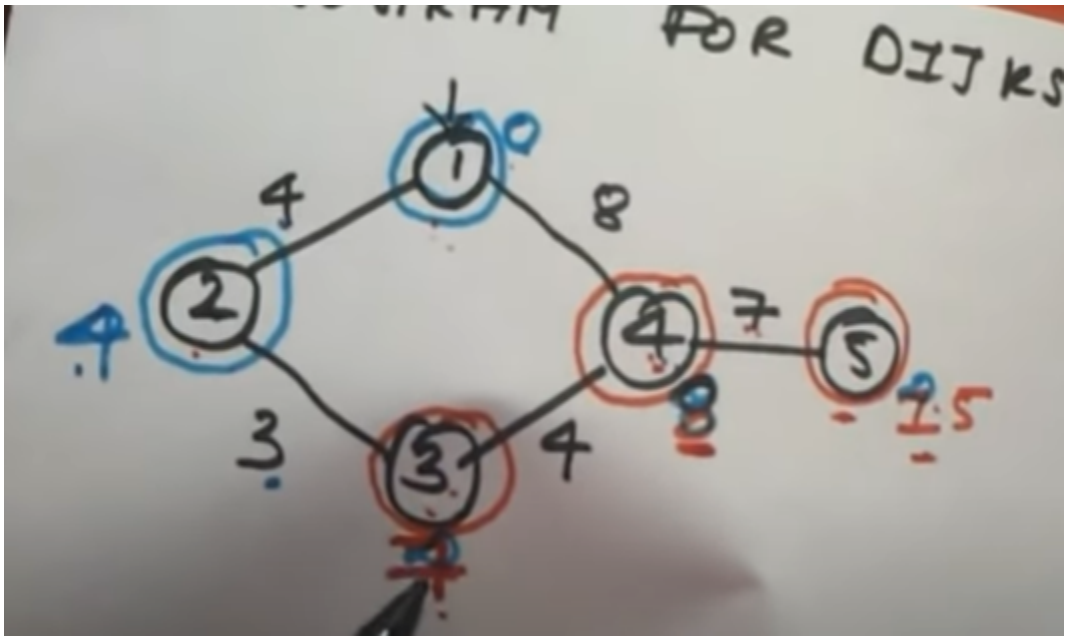
Week-3 Sliding window protocol

```
#include <stdio.h>

int main()
{
    int w,i,f,frames[50];
    printf("Enter window size: ");
```

```
    scanf("%d",&w);
    printf("Enter no of frames: ");
    scanf("%d",&f);
    printf("Enter %d frames : \n",f);

    for(i=1;i<=f;i++){
        scanf("%d",&frames[i]);
    }
    printf("\n\n");
    printf("With sliding window protocol the frames will be sent in the following
manner(assuming no corruption of frames)\n");
    printf("After sending %d frames at each stage sender waits for acknowledgement sent by
the reciever\n",w);

    for(i=1;i<=f;i++){
        if(i%w == 0){
            printf("%d ",frames[i]);
            printf("\nAcknowledgement of above frames sent is recieved by sender\n\n");
        }else{
            printf("%d ",frames[i]);
        }
    }
    if(f%w!=0){
        printf("\nAcknowledgement of above frames sent is recieved by sender\n\n");
    }

    return 0;
}
```

Week-4 Dijsktra's algorithm

C Program (Dijkstra's Algorithm)




```
#include <stdio.h>

int main()
{
    int n,i,j,p,v;
    printf("Enter no. of nodes: ");
    scanf("%d",&n);
    int cost[n+1][n+1];
    printf("Enter cost matrix: \n");
    for(i=1;i<=n;i++){
        for(j=1;j<=n;j++){
            scanf("%d",&cost[i][j]);
        }
    }
    printf("Enter end node(v): ");
    scanf("%d",&v);
    printf("Enter no. of paths: ");
    scanf("%d",&p);
    int path[p+1][n+1],dist[p+1];
    printf("Enter path matrix: \n");
    for(i=1;i<=p;i++){
        for(j=1;j<=n;j++){
            scanf("%d",&path[i][j]);
        }
    }
    int row,col;
    for(i=1;i<=p;i++){
        dist[i]=0;
```

```
        row=1;
        for(j=1;j<=5;j++){
            if(row!=v){
                col = path[i][j+1];
                dist[i] = dist[i]+cost[row][col];
            }
            row=col;
        }
    }
    int min = dist[1], index;
    for(i=1;i<=p;i++){
        if(dist[i]<=min){
            min = dist[i];
            index = i;
        }
    }
    printf("The min dist is %d\n",min);
    printf("min dist path:\n");
    for(i=1;i<=n;i++){
        if(path[index][i]!=0){
            printf("%d->",path[index][i]);
        }
    }

    return 0;
}

/*
INPUT:

5
0 4 0 8 0
4 0 3 0 0
0 3 0 4 0
8 0 4 0 7
0 0 0 7 0
5
2
1 2 3 4 5
1 4 5 0 0

*/
```

Week-5 broadcast tree for the subnet

```
#include<stdio.h>
int p,q,u,v,n;
int min=99,mincost=0;
int t[50][2],i,j;
int parent[50],edge[50][50];

void sunion(int l,int m){
    parent[l]=m;
}
int find(int l){
    if(parent[l]>0){
        l=parent[l];
    }
    return l;
}

void main(){
    printf("Enter the number of nodes: ");
    scanf("%d",&n);
    for(i=0;i<n;i++){
        printf("%c\t",65+i);
        parent[i]=-1;
    }
    printf("\n");
    for(i=0;i<n;i++){
        printf("%c ",65+i);
        for(j=0;j<n;j++){
            scanf("%d",&edge[i][j]);
        }
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            if(edge[i][j]!=99){
                if(min > edge[i][j]){
                    min = edge[i][j];
                    u=i;
                    v=j;
                }
            }
        }

    p=find(u);
    q=find(v);
    if(p!=q){
        t[i][0] = u;
        t[i][1] = v;
        mincost += edge[u][v];
```

```
        sunion(p,q);
    }else{
        t[i][0] = -1;
        t[i][1] = -1;
    }
    min =99;
    }
    printf("Minimum cost id %d\nMinimum Spanning tree is\n",mincost);
    for(i=0;i<n;i++){
        if(t[i][0]!=-1 && t[i][0]!=-1){
            printf("%c %c %d\n",65+t[i][0],65+t[i][1],edge[t[i][0]][t[i][1]]);
        }
    }
}

/*
INPUT:
3
1 2 3
1 2 3
4 5 6

*/
```

Week-6 Distance Vector Routing

```
#include <stdio.h>
struct node
{
    unsigned dist[20];
    unsigned from[20];
} rt[10];

int main()

{
    int dmat[20][20];
    int n, i, j, k, count = 0;
    printf("\nEnter the number of nodes : ");
    scanf("%d", &n);
    printf("\nEnter the cost matrix :\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
        {
            scanf("%d", &dmat[i][j]);
            dmat[i][i] = 0;
            rt[i].dist[j] = dmat[i][j];
```

```
            rt[i].from[j] = j;
        }

    do

    {

        count = 0;
        for (i = 0; i < n; i++)
            for (j = 0; j < n; j++)
                for (k = 0; k < n; k++)
                    if (rt[i].dist[j] > dmat[i][k] + rt[k].dist[j])

                    {
                        rt[i].dist[j] = rt[i].dist[k] + rt[k].dist[j];
                        rt[i].from[j] = k;
                        count++;
                    }

    } while (count != 0);
    for (i = 0; i < n; i++)
    {

        printf("\n\nState value for router %d is \n", i + 1);
        for (j = 0; j < n; j++)
        {

            printf("\t\nnode %d via %d Distance%d", j + 1, rt[i].from[j] + 1, rt[i].dist[j]);
        }
    }

    printf("\n\n");
}
/*
3
0 2 3
2 0 5
3 5 0
*/
```

Week-7 Data Encryption and Data Decryption

```
/*Week-7 Data encryption and data decryption */
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void main ()
{
   int i, ch, lp;
   char cipher[50], plain[50];
   char key[50];
   while (1)
   {
      printf ("\n\n-----MENU-----\n");
      printf ("\n1:Data Encryption\n2:Data Decryption\n3:Exit");
      printf ("\n\nEnter your choice:");
      scanf ("%d", &ch);
      switch (ch)
      {
         case 1:
            printf ("\nData Encryption");
            printf ("\nEnter the plain text:");
            scanf("%s",&plain);
            printf ("\nEnter the encryption key:");
            scanf("%s",&key);
            lp = strlen (key);
            for (i = 0; plain[i] != '\0'; i++)
            cipher[i] = plain[i] ^ lp;
            cipher[i] = '\0';
            printf ("\nThe encrypted text is:");
            for(i=0;cipher[i]!='\0';i++)
            printf("%c",cipher[i]);
            break;
         case 2:
            printf ("\nData decryption");
            for (i = 0; cipher[i] != '\0'; i++)
            plain[i] = cipher[i] ^ lp;
            printf ("\nDecrypted text is:");
            for(i=0;plain[i]!='\0';i++)
            printf("%c",plain[i]);
            break;
         case 3:
            exit (0);
      }
   }

}
```

```
/*

input:
1
computer
1234
2
3
*/
```

Week-8 Leaky bucket algorithm

leaky bucket algorithm | congestion control | networking | Bhanu Priya

Program 11: C/C++ Program for Congestion control using Leaky Bucket Algorithm

Congestion that occurs when a network node or link is carrying more data than it can handle

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

struct packet{
    int atime;
    int size;
}p[50];

int main(){
    int i,n,m,k=0;
    int bsize,bfilled,outrate;
    printf("Enter the number of packets: ");
    scanf("%d",&n);
    printf("Enter packets in the order of their arrival time \n");
    for(i=0;i<n;i++){
        printf("ENter the time and size: ");
        scanf("%d %d",&p[i].atime,&p[i].size);
    }
    printf("Enter the bucket size: ");
    scanf("%d",&bsize);
    printf("Enter the output rate: ");
    scanf("%d",&outrate);
    m=p[i-1].atime;
    i=1;
    k=0;
    bfilled=0;
    while(i<=m || bfilled!=0){
```

```
        printf("\n\nAt time %d",i);
        if(p[k].atime == i){
            if(bsize >= bfilled + p[k].size){
                bfilled +=p[k].size;
                printf("\n%d byte packet is inserted",p[k].size);
                k++;
            }else{
                printf("\n%d byte packet is discarder",p[k].size);
                k++;
            }
        }
        if(bfilled == 0){
            printf("\n No packet to transmitte");
        }else if(bfilled >= outrate){
            bfilled -= outrate;
            printf("\n%d bytes transfered",outrate);
        }else{
            printf("\n%d bytes transfered",bfilled);
            bfilled=0;
        }
        printf("\nPackets in the bucket %d byte",bfilled);
        i++;
    }
    return 0;
}
```

Week-9 Frame Sorting Technique used in Buffers

```
#include <stdio.h>
#include<stdlib.h>
struct frame{
    int fslno;
    char finfo[20];
}arr[10];
int n;

void sort(){
    int i,j,ex;
    struct frame temp;
    for(i=0;i<n;i++){
        for(j=i+1;j<n;j++){
            if(arr[i].fslno > arr[j].fslno){
                temp = arr[i];
                arr[i] = arr[j];
```

```
            arr[j] = temp;
            ex++;
          }

       }
    }
}
int main()
{
    int i;
    printf("Enter no. of frames: ");
    scanf("%d",&n);
    for(i=0;i<n;i++){
        arr[i].fslno = rand() % (50);
        printf("Enter the frame contents for sequence number %d : ",arr[i].fslno);
        scanf("%s",arr[i].finfo);

    }
    sort();
    printf("\n The frames in sequence \n");
    for(i=0;i<n;i++){
        printf("\n%d\t%s",arr[i].fslno,arr[i].finfo);
    }
    return 0;
}
```