BIT STUFFING

```
#include<stdio.h>
#include<math.h>
#include<string.h>

main()
{
        int i=0,j=0,k=0,cnt=0;
        char msg[100],stmsg[100],dstmsg[100],flag[10]="01111110";

        clrscr();

        printf("\nEnter the bit stream:");
        scanf("%s",msg);

        /*stuffing*/

        for(i=0;flag[i]!='\0';i++)
        {
                stmsg[i]=flag[i];
        }

        for(j=0;msg[j]!='\0';j++)
        {
```

```
        if(msg[j]=='1')
        {
                ++cnt;
                stmsg[i++]=msg[j];

                if(cnt==5)
                {
                        stmsg[i++]='0';
                        cnt=0;
                }
        }

        else
        {
                stmsg[i++]=msg[j];
                cnt=0;
        }
}

for(k=0;flag[k]!='\0';k++)
{
        stmsg[i++]=flag[k];
}
```

```
stmsg[i]='\0';

printf("\nStuffed bit stream is : %s",stmsg);

/*destuffing*/

cnt=0;
k=0;

for(j=8;j<i-8;j++)
{
        if(stmsg[j]=='1')
        {
                ++cnt;
                dstmsg[k++]=stmsg[j];

                if(cnt==5)
                {
                        j++;
                        cnt=0;
                }
        }

        else
```

```
                {
                        dstmsg[k++]=stmsg[j];
                        cnt=0;
                }
        }

        dstmsg[k]='\0';

        printf("\nDestuffed message is : %s",dstmsg);

        getch();
}
```

OUTPUT

Enter the bit stream:0111101111100

Stuffed bit stream is : 011111100111101111100001111110

Destuffed message is : 0111101111100

Enter the bit stream:00011111000001

Stuffed bit stream is : 011111100001111100000101111110

Destuffed message is : 00011111000001

CHARACTER STUFFING

```
#include<stdio.h>
#include<math.h>
#include<string.h>

main()
{
        int i=0,j=0,k=0,l=0;
        char msg[100],stmsg[100],dstmsg[100],stx,etx,dle;

        clrscr();

        printf("\nEnter the starting element of text:");
        stx=getche();
        printf("\nEnter the data link enable text:");
        dle=getche();
        printf("\nEnter the ending element of text:");
        etx=getche();
        printf("\nEnter the mesage:");

        gets(msg);

        /*Stuffing*/
```

```
stmsg[j++]=dle;
stmsg[j++]=stx;

for(i=0;msg[i]!='\0';i++)
{
        stmsg[j++]=msg[i];

        if(msg[i]==stx || msg[i]==etx || msg[i]==dle)
        {
                stmsg[j++]=msg[i];
        }
}

stmsg[j++]=dle;
stmsg[j++]=etx;
stmsg[j]='\0';

printf("\nStuffed message is:");
puts(stmsg);

/*destuffing*/

for(k=2;k<j-2;k++)
```

```
        {
                dstmsg[l++]=stmsg[k];

                if(stmsg[k]==stx || stmsg[k]==etx || stmsg[k]==dle)
                {
                        k++;
                }
        }

        dstmsg[l]='\0';

        printf("\nDestuffed message is:");
        puts(dstmsg);

        getch();
}
```

OUTPUT

Enter the starting element of text:a

Enter the data link enable text:b

Enter the ending element of text:c

Enter the mesage:welcome

Stuffed message is:bawelccomebc

Destuffed message is:welcome

Enter the starting element of text:adm

Enter the data link enable text:bdk

Enter the ending element of text:cjl

Enter the mesage:karnataka

Stuffed message is:bdkadmkarnatakabdkcjl

Destuffed message is:karnataka

DISTANCE VECTOR ALGORITHM

```
#include<stdio.h>
struct node
{
	unsigned dist[20];
	unsigned from[20];
}rt[10];
int main()
{
	int costmat[20][20];
	int nodes,i,j,k,count=0;
	printf("\nEnter the number of nodes : ");
	scanf("%d",&nodes);//Enter the nodes
	printf("\nEnter the cost matrix :\n");
	for(i=0;i<nodes;i++)
	{
		for(j=0;j<nodes;j++)
		{
			scanf("%d",&costmat[i][j]);
			costmat[i][i]=0;
			rt[i].dist[j]=costmat[i][j];//initialise the distance equal to cost matrix
			rt[i].from[j]=j;
```

```
        }


    }
        do
        {
            count=0;
            for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance from the node i to k using the cost matrix
            //and add the distance from k to node j
            for(j=0;j<nodes;j++)
            for(k=0;k<nodes;k++)
                if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
                {//We calculate the minimum distance
                    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<nodes;i++)
        {
            printf("\n\n For router %d\n",i+1);
            for(j=0;j<nodes;j++)
            {
```

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


        }
    printf("\n\n");
    getch();
}
```

OUTPUT

Enter the number of nodes : 3

Enter the cost matrix :

0 1 5

1 0 2

5 2 0

For router 1

node 1 via 1 Distance 0

node 2 via 2 Distance 1

node 3 via 2 Distance 3

For router 2

node 1 via 1 Distance 1

node 2 via 2 Distance 0

node 3 via 3 Distance 2

For router 3

node 1 via 2 Distance 3

node 2 via 2 Distance 2

node 3 via 3 Distance 0

DJKSTRA'S ALGORITHM

```
#include<stdio.h>
#include<math.h>
#include<ctype.h>
#include<string.h>
#include<conio.h>

main()
{
        struct
        {
                int pred,len;
                char lab;
        }nodeinfo[20];

        int n,i,j,k,min,inf=9999,src,dst;
        int net[5][5];

        clrscr();

        printf("\nEnter the no: of nodes:");
        scanf("%d",&n);
```

```
 printf("\nEnter the cost matrix:");


for(i=1;i<=n;i++)
{
        for(j=1;j<=n;j++)
        {
                if(i<j)
                {
                        printf("\nCost b/w nodes %d to %d is:",i,j);
                        scanf("%d",&net[i][j]);
                        net[j][i]=net[i][j];
                        net[i][i]=0;
                        net[j][j]=0;
                }
        }
}

printf("\nEnter the src: node:");
scanf("%d",&src);
printf("\nEnter the dst: node:");
scanf("%d",&dst);

for(i=1;i<=n;i++)
```

```
{
        nodeinfo[i].pred=0;
        nodeinfo[i].len=9999;
        nodeinfo[i].lab='t';
}

k=dst;
nodeinfo[dst].len=0;
nodeinfo[dst].lab='p';

do
{
        for(i=1;i<=n;i++)
        {
                if( (nodeinfo[i].lab=='t') && (net[k][i]!=0) )
                {
                        if( (net[k][i]+nodeinfo[k].len)<nodeinfo[i].len )
                        {
                                nodeinfo[i].len=nodeinfo[k].len+net[k][i];
                                nodeinfo[i].pred=k;
                        }
                }
        }
```

```
            min=inf;

        for(i=1;i<=n;i++)
        {
                if( (nodeinfo[i].lab=='t') && (nodeinfo[i].len<min) )
                {
                        k=i;
                        min=nodeinfo[i].len;
                }
        }

        nodeinfo[k].lab='p';
}while(k!=src);

printf("\nCost b/w src: & dst: is : %d",nodeinfo[src].len);

printf("\nShortest path b/w src: & dst: is : ");

do
{
        printf("\n%d to %d",k,nodeinfo[k].pred);
        k=nodeinfo[k].pred;
}while(k!=dst);
```

```
        getch();
}
```

OUTPUT

Enter the no: of nodes:4

Enter the cost matrix:

Cost b/w nodes 1 to 2 is:2

Cost b/w nodes 1 to 3 is:8

Cost b/w nodes 1 to 4 is:3

Cost b/w nodes 2 to 3 is:2

Cost b/w nodes 2 to 4 is:10

Cost b/w nodes 3 to 4 is:3

Enter the src: node:1

Enter the dst: node:3

Cost b/w src: & dst: is : 4

Shortest path b/w src: & dst: is :

1 to 2

2 to 3

CRC-CCITT POLYNOMIAL

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
#include<ctype.h>
#include<string.h>

main()
{
        char msg[100],gen[100],app[100],azero[100];
        int i,j,lmsg,lgen,lapp,lappr;

        clrscr();

        printf("\nEnter the bit stream:");
        scanf("%s",msg);
        printf("\nEnter the generator:");
        scanf("%s",gen);

        lgen=strlen(gen);
        lmsg=strlen(msg);
```

```
 for(i=0;i<(lgen-1);i++)
   azero[i]='0';
azero[i]='\0';

strcat(msg,azero);
strcpy(app,msg);

lapp=strlen(app);

for(i=0;i<=(lapp-lgen);i++)
{
        if(app[i]=='1')
        {
                for(j=0;j<lgen;j++)
                {
                        if(app[i+j]==gen[j])
                                app[i+j]='0';
                        else
                                app[i+j]='1';
                }
        }
}
```

```
 printf("\nChecksum is : ");

for(i=lmsg;i<(lmsg+lgen-1);i++)
        printf("%c",app[i]);

for(i=lmsg;i<(lmsg+lgen);i++)
        msg[i]=app[i];

printf("\nChecksum appended bit stream is : %s",msg);

printf("\nEnter the received bit stream:");
scanf("%s",app);

lappr=strlen(app);

if(lappr!=lapp)
        printf("\nReceived bit stream is in error");

else
{
        for(i=0;i<=(lapp-lgen);i++)
        {
```

```
            if(app[i]=='1')
        {
                for(j=0;j<lgen;j++)
                {
                        if(app[i+j]==gen[j])
                                app[i+j]='0';
                        else
                                app[i+j]='1';
                }
        }
}

j=0;

for(i=0;app[i]!='\0';i++)
{
        if(app[i]=='1')
                j++;
}

if(j!=0)
```

```
                printf("\nReceived bt stream is errorneous:");



        else
                printf("\nReceived bit stream is free of error:");
    }

    getch();
}
```

OUTPUT

Enter the bit stream:100110011_ _

Enter the generator:101

Checksum is : 00

Checksum appended bit stream is : 1001100100

Enter the received bit stream:1001100100

Received bit stream is free of error:

Enter the bit stream: 11111_ _0000

Enter the generator: 110

Checksum is : 00

Checksum appended bit stream is : 1111000000

Enter the received bit stream:111100000

Received bit stream is in error

STOP AND WAIT ALGORITHM

```
#include <stdio.h>
#include <stdlib.h>
#define RTT 4
#define TIMEOUT 4
#define TOT_FRAMES 7

enum {NO,YES} ACK;

int main()
{
        int wait_time,i=1;
        ACK=YES; // initially take ACK as YES to send first frame

        for(;i<=TOT_FRAMES;)
        {
                if (ACK==YES && i!=1) // if i is not 1st frame and ACK=YES
                {
                        printf("\nSENDER: ACK for Frame %d Received.\n",i-1);
                }

                printf("\nSENDER: Frame %d sent, Waiting for ACK...\n",i);
```

```
                ACK=NO; // after sending i_th frame set ACK=NO

                wait_time= rand() % 4+1; //generate random wait time betn 1 to 4

                if (wait_time==TIMEOUT)// resend the frame
                {
                        printf("SENDER: ACK not received for Frame %d=>TIMEOUT Resending
Frame...",i);
                }

                else
                {
                        sleep(RTT/2); // wait for RTT/2
                        printf("\nRECEIVER: Frame %d received,ACK sent\n",i);
printf("-----------------------------------------");
                        ACK=YES;//    set ACK=YES
                        sleep(RTT/2); // wait for RTT/2
                        i++; // select next frame
                }
        }

        return 0;
}
```

OUTPUT

SENDER: Frame 1 sent, Waiting for ACK...

SENDER: ACK not received for Frame 1=>TIMEOUT Resending Frame...

SENDER: Frame 1 sent, Waiting for ACK...

RECEIVER: Frame 1 received,ACK sent

SENDER: ACK for Frame 1 Received.

SENDER: Frame 2 sent, Waiting for ACK...

RECEIVER: Frame 2 received,ACK sent

SENDER: ACK for Frame 2 Received.

SENDER: Frame 3 sent, Waiting for ACK...

SENDER: ACK not received for Frame 3=>TIMEOUT Resending Frame...

SENDER: Frame 3 sent, Waiting for ACK...

RECEIVER: Frame 3 received,ACK sent

SENDER: ACK for Frame 3 Received.

SENDER: Frame 4 sent, Waiting for ACK...

SENDER: ACK not received for Frame 4=>TIMEOUT Resending Frame...

SENDER: Frame 4 sent, Waiting for ACK...

RECEIVER: Frame 4 received,ACK sent

SENDER: ACK for Frame 4 Received.

SENDER: Frame 5 sent, Waiting for ACK...

RECEIVER: Frame 5 received,ACK sent

SENDER: ACK for Frame 5 Received.

SENDER: Frame 6 sent, Waiting for ACK...

RECEIVER: Frame 6 received,ACK sent

SENDER: ACK for Frame 6 Received.

SENDER: Frame 7 sent, Waiting for ACK...

RECEIVER: Frame 7 received,ACK sent

SLIDING WINDOW

```
#include <stdio.h>    p
#include <stdlib.h>
#define RTT 5

int main()
{
        int window_size,i,f,frames[50];

        printf("Enter window size: ");
        scanf("%d",&window_size); // read window size

        printf("\nEnter number of frames to transmit: ");
        scanf("%d",&f); // read no. of frames

        printf("\nEnter %d frames: ",f);

        for(i=1;i<=f;i++)
                scanf("%d",&frames[i]); //read frame values

        printf("\nAfter sending %d frames at each stage sender waits for ACK ",window_size);
        printf("\nSending frames in the following manner....\n\n");
```

```
        for(i=1;i<=f;i++)
{
          if(i%window_size!=0) // collect the frames to fit in window
          {
                    printf(" %d",frames[i]);
          }

          else
          {
                    printf(" %d\n",frames[i]); // send the frames
                    printf("SENDER:waiting for ACK...\n\n");

                    sleep(RTT/2); // wait for RTT/2
                    printf("RECEIVER:Frames Received, ACK Sent\n");
                    printf("----------------------------------------\n");

                    sleep(RTT/2); // wait for RTT/2
                    printf("SENDER:ACK received, sending next frames\n");

          }


}
```

```
    if(f%window_size!=0) // send the left over frames
    {
        printf("\nSENDER:waiting for ACK...\n");

        sleep(RTT/2);// wait for RTT/2

        printf("\nRECEIVER:Frames Received, ACK Sent\n");
        printf("-----------------------------------------------\n");

        sleep(RTT/2); // wait for RTT/2
        printf("SENDER:ACK received.");
    }

    return 0;
}
```

OUTPUT

Enter window size: ^[[F_ __ __ __ _1

Enter number of frames to transmit: 4

Enter 4 frames:

11

22

33

44

After sending 1 frames at each stage sender waits for ACK

Sending frames in the following manner....

11

SENDER:waiting for ACK...

RECEIVER:Frames Received, ACK Sent

SENDER:ACK received, sending next frames

22

SENDER:waiting for ACK...

RECEIVER:Frames Received, ACK Sent

SENDER:ACK received, sending next frames

33

SENDER:waiting for ACK...

RECEIVER:Frames Received, ACK Sent

SENDER:ACK received, sending next frames

44

SENDER:waiting for ACK...

RECEIVER:Frames Received, ACK Sent

SENDER:ACK received, sending next frames