Write a program for HLDC frame to perform Bit Stuffing

```
#include <stdio.h>
int main()
{
int i=0, count=0;
char a[100];
printf("Enter the Bits:");
scanf("%s",a);
printf("\n After Bit Stuffing \n");
printf("01111110");
for(i=0;a[i];i++)
{
if(a[i]=='1')
count++;
else
count=0;
printf("%c",a[i]);
if(count==5)
{
printf("0");
count=0;
}
}
printf("01111110");
return 1;
}
```

Result:

Enter the Bits: 000111111100111110

After Bit Stuffing
01111110000111110110011111000111110

Write a program for HLDC frame to perform Character(Byte) Stuffing

```
#include<stdio.h>
#include<string.h>
int main()
{
char str[50],dest[50]=" ";
int i;
printf("Enter the character string :");
gets(str);
printf("%s",str);
printf("\n Original data:\n");
puts(str);
strcat(dest,"escstx");
for(i=0;i<strlen(str);)
{
if((str[i]=='e' && str[i+1]=='s' && str[i+2]=='c'))
{
strcat(dest,"escesc");
i=i+3;
}
else
{
char temp [2];
temp[0]=str[i];
temp[1]='\0';
strcat(dest,temp);
i++;
}
}
strcat(dest,"escdtx");
printf("\n Stuffed Data:\n");
puts(dest);
}
```

Result:

```
Enter the character string :hello vdit esc cn lab
hello vdit esc cn lab
 Original data:
hello vdit esc cn lab

 Stuffed Data:
 escstxhello vdit escesc cn labescdtx
```

Write a program for distance vector algorithm to find suitable path for transmission

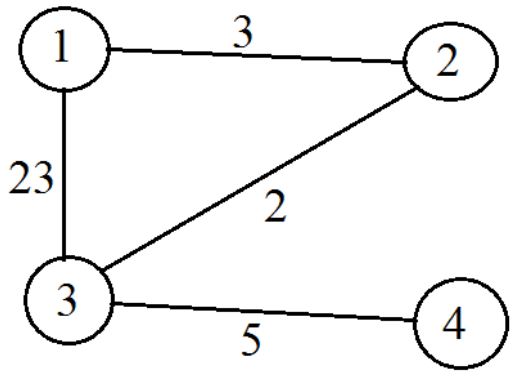
```
#include<stdio.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}
rt[10];
int main()
{
int costmat[20][20];
int nodes,i,j,k,count=0;
printf("\nEnter the number of nodes : ");
scanf("%d",&nodes);
printf("\nEnter the cost matrix :\n");
for(i=0;i<nodes;i++)
{
for(j=0;j<nodes;j++)
{
scanf("%d",&costmat[i][j]);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

Result:

Enter the number of nodes : 4

Enter the cost matrix :
0 3 23 999
3 0 2 999
23 2 0 5
999 999 5 0



For router 1

node 1 via 1 Distance 0
node 2 via 2 Distance 3
node 3 via 2 Distance 5
node 4 via 3 Distance 10

For router 2

node 1 via 1 Distance 3
node 2 via 2 Distance 0
node 3 via 3 Distance 2
node 4 via 3 Distance 7

For router 3

node 1 via 2 Distance 5
node 2 via 2 Distance 2
node 3 via 3 Distance 0
node 4 via 4 Distance 5

For router 4

node 1 via 3 Distance 10
node 2 via 3 Distance 7
node 3 via 3 Distance 5
node 4 via 4 Distance 0

For the given data, use CRC-CCITT polynomial to obtain CRC code

```
#include<stdio.h>
#include<string.h>
#define N strlen(g)

char t[128],cs[128],g[]="1101";
int a,e,c;
void xor()
{
for(c=1;c<N;c++)
cs[c]=((cs[c]==g[c])?'0':'1');
}
void crc()
{
for(e=0;e<N;e++)
cs[e]=t[e];
do
{
if(cs[0]=='1')
xor();
for(c=0;c<N-1;c++)
{
cs[c]=cs[c+1];
}
cs[c]=t[e++];
}
while(e<=a+N-1);
}

void main()
{
printf("\nEnter the polynomial:");
scanf("%s",t);
printf("\nGenerator polynomial is:%s",g);
a=strlen(t);
for(e=a;e<a+N-1;e++)
t[e]='0';
printf("\nModified t[u] is: %s",t);
crc();
printf("\nChecksum is:%s",cs);
for(e=a;e<a+N-1;e++)
t[e]=cs[e-a];


printf("\nFinal codeword is:%s",t);
```

```
printf("\nTest error detection 0(yes) 1(no) ?:");
scanf("%d",&e);
if(e==0)
{
printf("enter the position where the error is to be inserted:");
scanf("%d",&e);
e=e-1;
t[e]=(t[e]=='0')?'1':'0';
printf("Errorneous data: %s\n",t);
}
crc();
for(e=0;(e<N-1)&&(cs[e]!='1');e++) ;
if(e<N-1==1)
printf("Error detected.");
else
printf("No error detected.");
}
```

Result:

```
Enter the polynomial:1101

Generator polynomial is:1101
Modified t[u] is: 1101000
Checksum is:000
Final codeword is:1101000
Test error detection 0(yes) 1(no) ?:0
enter the position where the error is to be inserted:2
Erroneous data: 1001000
```

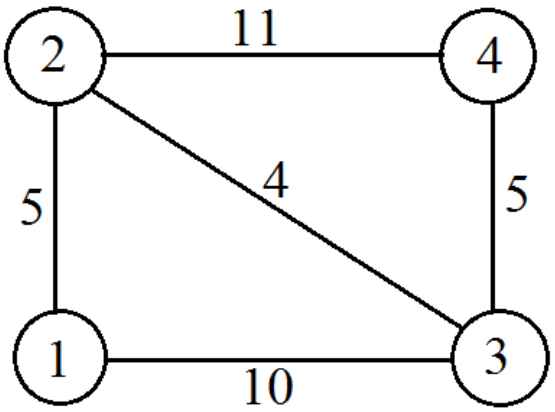
Implement Dijkstra's algorithm to compute the shortest routing path

```
#include<stdio.h> i
void dijikstras(int cost[10][10],int dist[10],int n,int v)
{
int i,u,w,count,flag[10],min;
for(i=1;i<=n;i++)
{
flag[i]=0;
dist[i]=cost[v][i];
}
flag[v]=1;
dist[v]=0;
count=2;
while(count<n)
{
for(i=1,min=999;i<=n;i++)
{
if(dist[i]<min && !flag[i])
{
min=dist[i];
u=i;
}
}
flag[u]=i;
count++;
for(w=1;w<=n;w++)
{
if(dist[u]+cost[u][w]<dist[w] && !flag[w])
dist[w]=dist[u]+cost[u][w];
}
}
}
int main()
{
int n,cost[10][10],source,i,j,dist[10];
printf("Enter the number of vertices\n");
scanf("%d",&n);
printf("Enter the cost matrix\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
{
```

```
scanf("%d",&cost[i][j]);
if(cost[i][j]==0)
cost[i][j]=999;
}
printf("source\n");
scanf("%d",&source);
dijikstras(cost,dist,n,source);
printf("Vertex \t Destination\t Cost\n");
for(i=1;i<=n;i++)
if(source!=i)
printf("%d\t\t%d\t%d\n",source,i,dist[i]);
}
```

Result:

```
Enter the number of vertices
4
Enter the cost matrix
0 5 10 999
5 0 4 11
10 4 0 5
999 11 5 0
source
1
Vertex   Destination     Cost
1            2      5
1            3      9
1            4      14
```



Implementation of Stop and Wait Protocol

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

void main()
{
int i,j,no_frames,x,x1=10,x2=2;
i=1;
j=1;
printf("Enter the no of frames:");
scanf("%d",&no_frames);
printf("\nNo. of Frame is %d", no_frames);
while(no_frames>0)
{
printf("\n Sending Frame %d",i);
srand(x1++);
x=rand()%10;
if(x%2==0)
{
printf("Waiting for %d seconds\n",x2);
sleep(x2);
printf("\nretransmitting frame %d",i);
srand(x1++);
x=rand()%10;
}
printf("\n ack for frame %d",j+1);
no_frames-=1;
i++;
j++;
}
printf("\nend of stop and wait protocols");

}
```

Result:

```
Enter the no of frames:7

No. of Frame is 7
 Sending Frame 1
 ack for frame 2
 Sending Frame 2
 ack for frame 3
 Sending Frame 3Waiting for 2 seconds
```

```
retransmitting frame 3
 ack for frame 4
 Sending Frame 4
 ack for frame 5
 Sending Frame 5
 ack for frame 6
 Sending Frame 6Waiting for 2 seconds

retransmitting frame 6
 ack for frame 7
 Sending Frame 7Waiting for 2 seconds

retransmitting frame 7
 ack for frame 8
end of stop and wait protocols
```

Implementation of Go-Back-N Protocol

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int n,r;
struct frame
{
char ack;
int data;
}
frm[10];

void main()

{
int i,j;
printf("\nEnter the no. of packets to be sent:");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("\nEnter data for packet%d :",i);
scanf("%d",&frm[i].data);
frm[i].ack='y';
}
rand();
r=rand()%n;
frm[r].ack='n';
for(i=1;i<=n;i++)
{
if(frm[i].ack=='n')
printf("\nThe packet number %d is not received\n",r);
}
printf("\n resending packet %d ",r);
for(i=r;i<=n;i++)
{
sleep(2);
frm[i].ack='y';
printf("\nReceived data of packet %d is %d",i,frm[i].data);
}
printf("\n all packets sent successfully\n");
}
```

Result:

Enter the no. of packets to be sent:5

Enter data for packet1 :7

Enter data for packet2 :36

Enter data for packet3 :15

Enter data for packet4 :62

Enter data for packet5 :11

The packet number 1 is not received

resending packet 1
Received data of packet 1 is 7
Received data of packet 2 is 36
Received data of packet 3 is 15
Received data of packet 4 is 62
Received data of packet 5 is 11
all packets sent successfully

Implementation congestion control using leaky bucket algorithm

```
/*Implementation of leaky bucket algorithm.*/
#include<stdio.h>
#include<stdlib.h>
void main()
{
int i,packets[10],content=0,newcontent,time,clk,bcktsize,oprate;
for(i=0;i<5;i++)
{
packets[i]=rand()%10;
if(packets[i]==0)
 --i;
}
printf("\n Enter output rate of the bucket: \n");
scanf("%d",&oprate);
printf("\n Enter Bucketsize\n");
scanf("%d",&bcktsize);
for(i=0;i<5;++i)
{
if((packets[i]+content)>bcktsize)
{
if(packets[i]>bcktsize)
printf("\n Incoming packet size %d greater than the size of the bucket\n",packets[i]);
else
printf("\n bucket size exceeded\n");
}
else
{
newcontent=packets[i];
content+=newcontent;
printf("\n Incoming Packet : %d\n",newcontent);
printf("\n Transmission left : %d\n",content);
time=rand()%10;
printf("\n Next packet will come at %d\n",time);
for(clk=0;clk<time && content>0;++clk)
{
printf("\n Left time %d",(time-clk));
sleep(1);
```

```
if(content)
{
printf("\n Transmitted\n");
if(content<oprate)
content=0;
else
content=content-oprate;
printf("\n Bytes remaining : %d\n",content);
}
else
printf("\n No packets to send\n");
}
}
]
}
```

Program Output

```
Enter output rate of the bucket:
5
 Enter Bucketsize
5
 Incoming Packet : 3
 Transmission left : 3
 Next packet will come at 5
 Left time 5
 Transmitted
 Bytes remaining : 0
 Incoming packet size 6 greater than the size of the bucket
 Incoming packet size 7 greater than the size of the bucket
 Incoming Packet : 5
 Transmission left : 5
 Next packet will come at 6
 Left time 6
 Transmitted
 Bytes remaining : 0
 Incoming Packet : 3
 Transmission left : 3
 Next packet will come at 2
 Left time 2
```

Transmitted
Bytes remaining : 0