1. Bit Stuffing

```
#include<stdio.h>

#include<string.h>

int main()

{

char a[20],fs[50]=" ",t[6],r[5];

int i,j,p=0,q=0;

printf("enter bit string:");

scanf("%s",&a);

strcat(fs,"01111110");

if(strlen(a)<5)

{

strcat(fs,a);

}

else

{

for(i=0;i<strlen(a)-4;i++)

{

for(j=i;j<i+5;j++)

{

t[p++]=a[j];

}

t[p]='\0';

if(strcmp(t,"11111")==0)

{

strcat(fs,"111110");

i=j-1;

}

else

{
```

```
r[0]=a[i];
r[2]='\0';
strcat(fs,r);
}
p=0;
}
for(q=i;q<strlen(a);q++)
{
t[p++]=a[q];
}
t[p]='\0';
strcat(fs,t);
}
strcat(fs,"01111110");
printf("after stuffing:%s",fs);
//getch();
}
```

2. Character Stuffing

```
#include<stdio.h>
#include<string.h>
int main()
{
char a[30],fs[50] =" ",t[3] ,sd[2] ,ed[3] ,x[3],s[3],d[3],y[3];
int i,j,p=0,q=0;
printf("\n enter the string to be stuffed:");
scanf("%s",&a);
printf("\n enter the character that represents starting delimiter:");
scanf("%s",&sd);
printf("\n enter the character that represents ending delimiter:");
scanf("%s",&ed);
x[0]=s[0]=s[1]=sd[0];
x[1]=s[2]=sd[1]='\0';
y[0]=d[0]=d[1]=ed[0];
d[2]=y[1]=ed[1]='\0';
strcat(fs,x);
printf("strlen[a]=%d",strlen(a));
for(i=0;i<strlen(a);i++)
{
t[0]=a[i];
t[1]='\0';
if(t[0]==sd[0])
strcat(fs,s);
else if(t[0]==ed[0])
strcat(fs,d);
else
strcat(fs,t);
}
strcat(fs,y);
printf("\n after byte stuffing:%s\n",fs);
//getch();
}
```

3. Leaky Bucket

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

#define min(x, y) ((x) < (y) ? (x) : (y))

int main()
```

```
{
    int orate, drop = 0, cap, x, y, 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 no of packets coming at second %d:", i + 1);
        scanf("%d", &inp[i]);
        i++;
        printf("\n enter 1 to continue or 0 to quit.......");
        scanf("%d", &ch);
    } while (ch);

    nsec = i;
    printf("\n \t second \t received \t sent \t dropped \t remainder \n");

    for (i = 0; count || i < nsec; i++)
    {
        printf("\t %d", i + 1);
        printf("\t %d\t", inp[i]);
        printf("\t \t %d", 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;
    }
```

4. Stop and Wait Protocol

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
int i=1,noframes,x,x1;
i=1;
printf("enter number of frames to be sent:");
scanf("%d",&noframes);
x=rand();
while(noframes>0)
{
printf("\n sending frame is:%d",i);
printf("\n random number is:%d",x);
if(x%2==0)
{
for(x1=1;x1<3;x1++)
{
printf("\n waiting for%d seconds\n",x1);
//sleep(1);
}
printf("/n resending frame%d\n",i);
}
printf("\n acknowledgement for frame%d\n",i);
noframes=noframes-1;
x=rand();
i++;
}
printf("\n end of stop and wait protocol");

}
```

5. Sliding Window Protocol

```
#include<stdio.h>
int main()
{
int i,w,f,frames[50];
printf("enter window size:");
scanf("%d",&w);
printf("enter no of frames to transmit:");
scanf("%d",&f);
printf("\n enter%d frames:",f);
for(i=1;i<=f;i++)
scanf("%d",&frames[i]);
printf("\n with 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 acknowledge 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("\n acknowledgement of above frames sent is received by sender\n");
}
```

6. CRC

```
#include<stdio.h>
#include<stdlib.h>
int a[100],b[100],i,j,k,len,count=0;
int gp[]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1};
//void divide();
int main()
{
void divide();
```

```
printf("\n enter the length of data frame:");
scanf("%d",&len);
printf("\n enter 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;
divide();
 for(i=0;i<len;i++)
 b[i]=b[i]^a[i];
 printf("\n data to be transmitted:");
 for(i=0;i<len;i++)
 printf("%2d",b[i]);
 printf("\n enter the received data");
 for(i=0;i<len;i++)
 scanf("%d",&a[i]);
 divide();
 for(i=0;i<len;i++)
 if(a[i]!=0)
 {
 printf("\n error in received data:");
 return 0;
  }
  printf("\n data received is error free");
   }
              void divide()
{
```

```
          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;
  }
}
```

7. 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("\n enter the no of nodess:");
scanf("%d",&nodes);
printf("\n enter the cost matrix:");
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 for router %d \n",i+1);
for(j=0;j<nodes;j++)
{
printf("\t \n node %d via %d distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
}
//getch();
}
printf("\n \n");
//getch();
```

```
}
```

8. Dijikstras Algorithm

```
#include<stdio.h>
void dijkstras(int cost[10][10],int dist[10],int n,int s)
{
int i,v,count=1,visited[10],min;
for(i=1;i<=n;i++)
{
visited[i]=0;
dist[i]=cost[s][i];
}
visited[s]=1;
dist[s]=0;
while(count<=n)
{
min=999;
for(i=1;i<=n;i++)
{
if(dist[i]<min&&!visited[i])
{
min=dist[i];
v=i;
}
}
visited[v]=1;
count++;
for(i=1;i<=n;i++)
{
if(dist[v]+cost[v][i]<dist[i])
dist[i]=dist[v]+cost[v][i];
}
}
}
void main()
{
int n,cost[10][10],source,i,j,dist[10];
printf("enter the no 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);
dijkstras(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]);
//getch();
}
```