```
#include <stdio.h>  // Bit stuffing

int main()
{
   int n,i,j,count = 0,a[30],b[30];
   printf("Enter the length of the frame : ");
   scanf("%d",&n);
   printf("Enter the frame in 1's and 0's : ");
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);
   for(i=0,j=0;i<n;i++,j++)
   {
      b[j]=a[i];
      if(a[i]==1)
      {
         count++;
         if(count==5)
         {
            b[++j]=0;
            count=0;
         }
      }
      else
         count=0;
   }
   printf("After bit stuffing :");
   for(i=0;i<j;i++)
      printf("%d",b[i]);
   return 0;
}
```

```
#include <stdio.h>  // character stufiing
#include <string.h>

int main()
{
   char inp[100], out[200];

   char se_dlt[100] = "DLESTX";
   int i = 0,j = 0; // i will be looping through the inp and j through the out
   printf("Here we are considering DLE as a delimiter\n");
   printf("Enter the Message to be sent : ");
   scanf("%s", inp);
   printf("The Message entered : %s\n");

   // Converting the given message into uppercase
   for( int k = 0; inp[k] != '\0'; k++ )
      if( inp[k] >= 'a' && inp[k] <= 'z' )
         inp[k] = inp[k] - 32;
   // Printing the message after converting into uppercase
   printf("Showing the message entered in uppercase : %s\n");



   if (strlen(inp) < 3)
   {
      strcpy(out, se_dlt); // Copy se_dlt to out
      strcat(out, inp);   // Concatenate inp to out
      strcat(out, "DLEETX"); // Concatenate "DLEETX" to out

      printf("The string after character stuff : %s\n", out);
   }
```

```
   else
   {
      //printf("Implement remaining\n");
      strcpy(out, se_dlt);
      j = 6;
      while(inp[i] != '\0')
      {
         if( inp[i] == 'D' && inp[i+1] == 'L' && inp[i+2] == 'E')
         {
            strcat(out,"DLEDLE"); // doing because we found delimiter i.e "DLE" in
original message
            i = i + 3; // Finished "DLE" in the input data.. It has to read a next
character in input, Now i will be standing 3 character ahead
            j = j + 6; // added an Extra "DLE" along with the input so j will be
standing 6 characters ahead
         }
         else
         {
            out[j] = inp[i];
            i++;
            j++;
         }
      }
      strcat(out,"DLEETX");
      printf("Message after character stuffing : %s",out);
   }
   return 0;
}
```

```
#include<stdio.h>  // CRC
int gen[4],genl,frl,rem[4];

void remainder_value(int fr[])
{
   int k,k1,i,j;
   for(k=0;k<frl;k++)
   {
      if(fr[k]==1)
      {
         k1=k;
         for(i=0,j=k;i<genl;i++,j++)
         {
            rem[i]=fr[j]^gen[i];
         }
         for(i=0;i<genl;i++)
         {
            fr[k1]=rem[i];
            k1++;
         }
      }
   }
}

void main()
{
   int i,j,fr[8],dupfr[11],recfr[11],tlen,flag;
   frl=8; genl=4;
   printf("Enter frame : ");
   for(i=0;i<frl;i++)
   {
      scanf("%d",&fr[i]);
      dupfr[i]=fr[i];
   }
   printf("Enter generator : ");
   for(i=0;i<genl;i++)
      scanf("%d",&gen[i]);
```

```
    tlen=frl+genl-1;
    for(i=frl;i<tlen;i++)
    {
       dupfr[i]=0;
    }
    remainder_value(dupfr);
    for(i=0;i<frl;i++)
    {
       recfr[i]=fr[i];
    }
    for(i=frl,j=1;j<genl;i++,j++)
    {
       recfr[i]=rem[j];
    }
    remainder_value(recfr);
    flag=0;
    for(i=0;i<4;i++)
    {
       if(rem[i]!=0)
          flag++;
    }
    if(flag==0)
    {
       printf("Frame received correctly");
    }
    else
    {
       printf("The received frame is wrong");
    }
}
```

```
#include <stdio.h>  // Dijkstra's Algorithm
int c[40][40],s[20],d[40],n,v;
void SP()
{
	int i,u,j,m,x,k;
	for(i=0;i<n;i++)
	{
		s[i]=0;
		d[i]=c[v][i];
	}
	for(i=1;i<n;i++)
	{
		m=999;
		x=0;
		for(k=0;k<n;k++)
		{
			if((d[k]<m)&&(s[k]==0))
			{
				m=d[k];
				x=k;
			}
		}
		u=x;
		s[u]=1;
		for(j=0;j<n;j++)
		{
			if(c[u][j]<999 && s[j]==0)
				if(d[j]>d[u]+c[u][j])
					d[j]=d[u]+c[u][j];
		}

	}
}
```

```
void main()
{
	int i,j;
	printf("Enter no.of vertices : \n");
	scanf("%d",&n);
	printf("Enter cost adjacency matrix : \n");
	for(i=0;i<n;i++)
		for(j=0;j<n;j++)
			scanf("%d",&c[i][j]);
	printf("\nEnter source vertex (vertices start from 0): ");
	scanf("%d",&v);

	SP();
	printf("\nthe shortest path from vertex %d to all vertices : ",v);
	for(i=0;i<n;i++)
	{
		if(d[i]<999)
			printf("\nNode %d to Node %d --> %d",v,i,d[i]);
		else
			printf("\nNode %d to Node %d --> inf",v,i);
	}
}
```

```
#include<stdio.h>  // Distance vector routing Algorithm

int c[20][20], d[20], s[20];
int n, v;

void DijkstraAlogorithm()
{
   int i, j, k, m, u, x;
   for ( i = 0; i < n; i++ )
   {
      s[i] = 0;
      d[i] = c[v][i];
   }

   for( i = 1; i < n; i++ )
   {
      m = 99;
      x = 0;
      for ( k = 0; k < n; k++ )
      {
         if( d[k] < m && s[k] == 0 )
         {
            m = d[k];
            x = k;
         }
      }
      u = x;
      s[u] = 1;
      for ( j = 0; j < n; j++ )
      {
         if ( c[u][j] < 99 && s[j] == 0 )
            if ( d[j] > d[u] + c[u][j] )
               d[j] = d[u] + c[u][j];
      }
   }
}
```

```
int main()
{
   int i, j;
   printf("Enter the no of nodes : ");
   scanf("%d", &n);
   printf("Enter the cost matix\n");
   for ( i = 0; i < n; i++ )
      for( j = 0; j < n; j++ )
         scanf("%d", &c[i][j]);

   for ( j = 0; j < n; j++ )
   {
      v = j;
      DijkstraAlogorithm();
      printf("\nState value for Router : %d \n", v);
      for( i = 0; i < n; i++ )
      {
         if ( d[i] < 99 )
            printf("Router %d to Router %d -> %d\n", v, i, d[i]);
         else
            printf("Router %d to Router %d -> infinity\n", v, i);
      }
   }
   return 0;
}
```