Character Count

```
#include<stdio.h>
#include<string.h>
void convertBinary(int c,char binary[10])//converting to binary
{
   int i;
   for(i=0;i<8;i++)
   {
      binary[i]='0';
   }
   binary[8]='\0';
   i=7;
   while(c)
   {
      binary[i--]=(c%2)+48;
      c=c/2;
   }
   return;
}
int main()
{
   char frames[25][25],bit[100]="",binary[100];
   int i,n,c;
   printf("Enter number of frames..:");
   scanf("%d",&n);
   for(i=0;i<n;)
   {
      printf("Enter %d frame : ",i+1);
      scanf("%s",frames[i]);
      if(strlen(frames[i])%8==0)
      {
         i++;
      }
      else
      {
         continue;
      }
   }
   for(i=0;i<n;i++)
   {
      c=(strlen(frames[i])/8)+1;//counting number of characters
      convertBinary(c,binary);
      strcat(bit,binary) ;//concating binary to res
      strcat(bit,frames[i]);//concating frame to res
   }
   printf("After performing the Character Count the result : %s",bit);//printing tranemitted frame
   return 0;
}
```

Output

Enter number of frames..:3

Enter 1 frame : 11110000

Enter 2 frame : 0111111011100010

Enter 3 frame : 11001100

After performing the Character Count the result : 000000101111000000000011011111101110001000000010110011 00

Character Stuffing

```
#include<stdio.h>
#include<string.h>
int main()
{
   int n,i,length,j,k,test=0,l,flag=0;
   printf("Enter Number Of Frames...:\n");
   scanf("%d",&n);
   char frames[25][25],buffer[10]="",dle[10],stx[10],etx[10],res[300]="";
   for(i=0;i<n;)
   {
      printf("Enter %d Frame : ",i+1);
      scanf("%s",frames[i]);
      if((strlen(frames[i])%8)!=0)//frame length should be multiple of 8
      {
         printf("Re-enter frames...:");
         continue;
      }
      i++;
   }
   do
   {
      printf("Enter dle...:");
      scanf("%s",dle);
      printf("Enter stx..:");
      scanf("%s",stx);
      printf("Enter etx..:");
      scanf("%s",etx);
      if(strlen(dle)==8&&strlen(stx)==8&&strlen(etx)==8)
      {
         flag=1;
      }
   }while(flag==0);
```

```
    for(i=0;i<n;i++)
    {
        test=0;
        l=0;
        strcat(res,dle);
        strcat(res,stx);
        length=strlen(frames[i])/8;
        for(j=0;j<length;j++)
        {
            for(k=test;k<=test+7;k++)
            {
                buffer[l++]=frames[i][k];
            }
            if(strcmp(buffer,dle)==0)//if flag appears in pattern
            {
                strcat(res,dle);//concat dle
                strcat(res,buffer);//concat buffer
            }
            else
            {
                strcat(res,buffer);//concat buffer
            }
            strcpy(buffer,"");//initializing buffer
            test+=8;
            l=00;
        }
        strcat(res,dle);
        strcat(res,etx);
    }
    printf("After Character Stuffing..:\n");
    printf("%s",res);
    return 0;
}
```

Output

Enter Number Of Frames...:3

Enter 1 Frame : 11110000

Enter 2 Frame : 0111111011100010

Enter 3 Frame : 11001100

Enter dle...:01111110

Enter stx..:10101010

Enter etx..:11100010

After Character Stuffing..:

01111110101010101111000001111110111000100111111010101010011111100111111011100010011111110111000101111111010101010101100110001111110111000101111111010101010110011000111111011100010

Bit Stuffing

```
#include<stdio.h>
#include<string.h>
int main()
{
   int n,i,c=0,j,k=0;
   printf("Enter Number Of Frames..:");
   scanf("%d",&n);
   char frames[25][25],bit[100]="",delimeter[10]="01111110";
   for(i=0;i<n;i++)
   {
      printf("Enter %d frame : ",i+1);
      scanf("%s",frames[i]);
   }
   for(i=0;i<n;i++)
   {
      c=0;//count of consecutive 1's
      strcat(bit,delimeter);
      k=strlen(bit);
      for(j=0;j<strlen(frames[i]);j++)
      {
         if(frames[i][j]=='0')
         {
            c=0;
            bit[k++]='0';
         }
         else
         {
            if(frames[i][j]=='1')
            {
               bit[k++]='1';
               c++;
            }
```

```
            if(c==5)//if consecutive 1's count is 5 then stuff a zero
            {
               bit[k++]='0';
               c=0;
            }
         }
      }
   }
   strcat(bit,delimeter);
   printf("After Bit Stuffing ..:");
   printf("%s",bit);
   return 0;
}
```

Output

```
Enter Number Of Frames..:3
Enter 1 frame : 1110010111
Enter 2 frame : 01111110
Enter 3 frame : 1001010
After Bit Stuffing ..:011111101110010111011111100111110100111111010010100111111 0
```

Cycle Redundancy Check

```
#include<stdio.h>
#include<string.h>
void Cyclic_Redundancy_Check(char [],char []);
void delete_element(char [],int);
int main()
{
   char dateword[100],polynomial[20]="";
   int choice,flag=0;
   printf("Enter Dateword..:");
   scanf("%s",dateword);
   printf("Enter your choice...:\n");
   printf("1 : CRC12\n");
   printf("2 : CRC16\n");
   printf("3 : CRC CCITT\n");
   printf("4 : user choice\n");
   scanf("%d",&choice);
   switch(choice)
   {
      case 1 : strcpy(polynomial,"1100000001111");break;
      case 2 : strcpy(polynomial,"11000000000000101");break;
      case 3 : strcpy(polynomial,"10001000000100001");break;
      case 4 : printf("Enter Polynomial..:\n");
            while(flag==0)
            {
              scanf("%s",polynomial);
                 if(polynomial[0]=='1'&&polynomial[strlen(polynomial)-1]=='1')
                 {
                    break;
                 }
              printf("Re-enter Ploynomial..:\n");
              flag=0;
           }
```

```
            break;
    }
    Cyclic_Redundancy_Check(dateword,polynomial);
    return 0;
}
void Cyclic_Redundancy_Check(char dateword[],char polynomial[])
{
    int length1,length2,i,j,operand1,operand2,result,k,t=0;
    char temp1[20],temp2[200];
    length1=strlen(polynomial);
    strcpy(temp2,dateword);
    length2=strlen(dateword);
    for(i=0,j=length2;i<length1-1;i++)//copying polynomial length-1 zeroes to string
    {
        temp2[j++]='0';
    }
    temp2[j]='\0';
    for(i=0;i<length1;i++)
    {
        temp1[i]=temp2[i];
    }
    temp1[i]='\0';
    for(i=strlen(temp1);i<=strlen(temp2);i++)//computing redundancy
    {
        if(temp1[0]=='1')
        {
            for(j=0;j<strlen(polynomial);j++)
            {
                operand1=temp1[j]-48;
                operand2=polynomial[j]-48;
                result=operand1^operand2;
                temp1[j]=result+48;
            }
        }
```

```
      else
      {
         operand2=0;
         for(j=0;j<strlen(temp1);j++)
         {
            operand1=temp1[j]-48;
            result=operand1^operand2;
            temp1[j]=result+48;
         }
      }
      k=strlen(temp1);
      temp1[k++]=temp2[i];
      temp1[k]='\0';
      delete_element(temp1,0);
   }
   printf("Remainder..:%s\n",temp1);
   strcat(dateword,temp1);//appending redundancy to dateword
   printf("Transmitted Bit..:%s\n",dateword);
   return;
}
void delete_element(char str[],int index)
{
   int i;
   for(i=index;i<strlen(str);i++)
   {
      str[i]=str[i+1];
   }
   str[i]='\0';
   return;
}
```

Output

Enter Dateword..:10101010

Enter your choice...:

1 : CRC12

2 : CRC16

3 : CRC CCITT

4 : user choice

2

Remainder..:0000001111111100

Transmitted Bit..:10101010000000111111100

Dijkstraw's Algorithim

```
#include<stdio.h>
#include<limits.h>
#define max 10
int n,adj_mat[max][max],distance;
struct node
{
   int predecessor;
   int length;
   enum {permanent,tentative} label;
};
int shortestPath(int source,int des,int path[])//computing shortest path using dijkstra's algorithm
{
   int i,j,min,k;
   struct node ele[n];
   for(i=0;i<n;i++){
      ele[i].predecessor=-1;
      ele[i].length=INT_MAX;
      ele[i].label=tentative;
   }
   ele[source].length=0;
   ele[source].label=permanent;
   k=source;
   do{
      for(i=0;i<n;i++){
         if((adj_mat[k][i]!=0)&&(ele[i].label==tentative))//making nodes as temporary
        {
            if(adj_mat[k][i]+ele[k].length<ele[i].length)
           {
               ele[i].predecessor=k;
               ele[i].length=ele[k].length+adj_mat[k][i];
            }
         }
```

```
    }
    k=0;
    min=INT_MAX;
    for(i=0;i<n;i++){
      if(ele[i].label==tentative&&ele[i].length<min)//computing smallest length node
      {
        min=ele[i].length;
        k=i;
      }
    }
    ele[k].label=permanent;//making node as permanent
  }while(k!=des);
  i=0;
  k=des;
  distance=ele[k].length;
  do{
    path[i++]=k;
    k=ele[k].predecessor;
  }while(k>=source);//storing path
  return i;
}
int main(){
  printf("Enter number of nodes..:\n");
  scanf("%d",&n);
  printf("Enter graph in adjacency matrix form..:\n");
  for(int i=0;i<n;i++){
    for(int j=0;j<n;j++){
      scanf("%d",&adj_mat[i][j]);
    }
  }
  char source,des;
  printf("Enter Source and Destination..:\n");
  scanf(" %c %c",&source,&des);
  int result=97;
```

```
    if(source>='A'&&source<='Z'){
        result=65;
    }
    char ip1[20],ip2[20];
    printf("Enter IP Address of source and destination..:\n");
    scanf("%s %s",ip1,ip2);
    int path[max];
    int ans=shortestPath(source-result,des-result,path);
    printf("The Shortest Path is..:");
    for(int i=ans-1;i>=0;i--){
        if(i==0){
            printf("%c\n",path[i]+result);
            continue;
        }
        printf("%c-->",path[i]+result);
    }
    printf("The forwarding tables are..:\n");
    printf("Destination Address\t\tOutput Interface\n");
    for(int i=ans-1;i>=0;i--){
        printf("%s\t\t\t%c\n",ip2,path[i]+result);
    }
    printf("Distance : %d\n",distance);
    return 0;
}
```

Output:

Enter number of nodes..:

5

Enter graph in adjacency matrix form..:

0 3 10000 8 7

3 0 1 4 10000

10000 1 0 2 10000

8 4 2 0 3

7 10000 10000 3 0

Enter Source and Destination..:

a

d

Enter IP Address of source and destination..:

123.654.789

258.369.147

The Shortest Path is..:a-->b-->c-->d

The forwarding tables are..:

Destination Address	Output Interface
258.369.147	a
258.369.147	b
258.369.147	c
258.369.147	d

Distance : 6

Distance Vector Routing Algorithm

```
#include<stdio.h>
struct node
{
   int dis[20];
   int from[20];
}rt[10];
int main()
{
   int n;
   printf("Enter Number Of Nodes : ");
   scanf("%d",&n);
   int costmat[n][n],i,j,k;
   printf("Enter Adjacency Matrix : \n");
   for(i=0;i<n;i++)
   {
      for(j=0;j<n;j++)
      {
         scanf("%d",&costmat[i][j]);//reading adjacency matrix
         costmat[i][i]=0;
         if(costmat[i][j]>100)
         {
            rt[i].from[j]=-1;
            rt[i].dis[j]=costmat[i][j];
            continue;
         }
         rt[i].from[j]=j;
         rt[i].dis[j]=costmat[i][j];
      }
   }
   int itr=1,count=0;
   printf("iteration : %d",itr++);
   for(i=0;i<n;i++)
```

```
{
   printf("\n\n For router %c \n",i+97);
   printf("\nDestination\tNext Hop\tDistance\n");
   for(j=0;j<n;j++)
   {
      printf("%c\t\t%c\t\t%d\n",j+97,rt[i].from[j]+97,rt[i].dis[j]);
   }
}
do
{
   count=0;
   printf("iteration : %d",itr++);
   for(i=0;i<n;i++)
   {
      for(j=0;j<n;j++)
      {
         for(k=0;k<n;k++)
         {
            if(rt[i].dis[j]>costmat[i][k]+rt[k].dis[j])//computing shortest path at  each routers
            {
               rt[i].dis[j]=costmat[i][k]+rt[k].dis[j];
               rt[i].from[j]=k;
               count=1;
            }
         }
      }
   }
   for(i=0;i<n;i++)
   {
      printf("\n\n For router %c \n",i+97);
      printf("\nDestination\tNext Hop\tDistance\n");
      for(j=0;j<n;j++)
      {
         printf("%c\t\t%c\t\t%d\n",j+97,rt[i].from[j]+97,rt[i].dis[j]);
```

```
        }
      }
   }while(count!=0);
   return 0;
}
```

Output

Enter Number Of Nodes : 4

Enter Adjacency Matrix :

0 2 10000 1

2 0 3 7

10000 3 0 11

1 7 11 0

iteration : 1

For router a

Destination	Next Hop	Distance
a	a	0
b	b	2
c	`	10000
d	d	1

For router b

Destination	Next Hop	Distance
a	a	2
b	b	0
c	c	3
d	d	7

For router c

Destination	Next Hop	Distance
a	`	10000
b	b	3
c	c	0
d	d	11

For router d

Destination	Next Hop	Distance
a	a	1
b	b	7
c	c	11
d	d	0

iteration : 2

For router a

Destination	Next Hop	Distance
a	a	0
b	b	2
c	b	5
d	d	1

For router b

Destination	Next Hop	Distance
a	a	2
b	b	0
c	c	3
d	a	3

For router c

Destination	Next Hop	Distance
a	b	5
b	b	3
c	c	0
d	b	6

For router d

Destination	Next Hop	Distance
a	a	1
b	a	3
c	a	6
d	d	0

iteration : 3

For router a

Destination	Next Hop	Distance
a	a	0
b	b	2
c	b	5
d	d	1

For router b

Destination	Next Hop	Distance
a	a	2
b	b	0
c	c	3
d	a	3

For router c

Destination	Next Hop	Distance
a	b	5
b	b	3
c	c	0
d	b	6

For router d

Destination	Next Hop	Distance
a	a	1
b	a	3
c	a	6
d	d	0

Broadcast Tree

```
#include<stdio.h>
#include<limits.h>
#include<string.h>
#define max 10
int n,adj_mat[max][max],finalPath[max][max];
struct node{
   int predecessor;
   int sucessor;
   int length;
   enum {permanent,tentative} label;
};
void shortestPath(int root,int path[])
{
   int i,j,min,k,count=0,prev;
   struct node ele[n];
   for(i=0;i<n;i++){
      ele[i].predecessor=-1;
      ele[i].sucessor=-1;
      ele[i].length=INT_MAX;
      ele[i].label=tentative;
   }
   ele[root].length=0;
   ele[root].label=permanent;
   k=root;
   do
   {
      prev=k;
      for(i=0;i<n;i++){
         if((adj_mat[k][i]!=0)&&(ele[i].label==tentative))
        {
            if(adj_mat[k][i]+ele[k].length<ele[i].length)
           {
```

```
                ele[i].predecessor=k;
                ele[i].length=ele[k].length+adj_mat[k][i];
            }
        }
    }
    k=0;
    min=INT_MAX;
    for(i=0;i<n;i++){
        if(ele[i].label==tentative&&ele[i].length<min)
        {
            min=ele[i].length;
            k=i;
        }
    }
    ele[k].label=permanent;//making node as permanent
    ele[ele[k].predecessor].sucessor=1;
    count++;
  }while(count<n);//computhing shortest to all node
  for(i=0;i<n;i++)
  {
    j=0;
    k=i;
    if(ele[i].sucessor==-1)
    {
        do
        {
            finalPath[i][j++]=k;
            k=ele[k].predecessor;
        }while(k>=0);
    }
    path[i]=j;
  }
  return;
}
```

```
int main(){
   printf("Enter number of nodes..:\n");
   scanf("%d",&n);
   printf("Enter graph in adjacency matrix form..:\n");
   for(int i=0;i<n;i++){
      for(int j=0;j<n;j++){
         scanf("%d",&adj_mat[i][j]);
      }
   }
   int root,j,path[max];
   printf("Enter root node..:\n");
   scanf("%d",&root);
   shortestPath(root,path);
   printf("The paths are : \n");
   for(int i=0;i<n;i++)//printing broadcast tree
   {
      if(path[i]>0)
      {
         for(int j=path[i]-1;j>=0;j--)
         {
            if(j==0)
            {
               printf("%d",finalPath[i][j]);
            }
            else
            {
               printf("%d-->",finalPath[i][j]);
            }
         }
         printf("\n");
      }
   }
   return 0;
}
```

Output

Enter number of nodes..:

5

Enter graph in adjacency matrix form..:

0 3 10000 8 7

3 0 1 4 10000

10000 1 0 2 10000

8 4 2 0 3

7 10000 10000 3 0

Enter root node..:

0

The paths are :

0-->1-->2-->3

0-->4