1] Write a Program to find the shortest path in a network of 6 to 10 nodes.

```
#include<iostream>
using namespace std;
class dj
{
 int n,cost[10][10],d[10],p[10],v[10];
 public: void read_matrix();
 void short_path(int);
 void display(int);
};
void dj::read_matrix()
{
 int i,j;
 cout<<"Enter the number of vertices\n";
 cin>>n;
 cout<<"Enter the cost adjacency matrix\n";
 for(i=0;i<n;i++)
     for(j=0;j<n;j++)
  cin>>cost[i][j];
}
void dj::short_path(int src)
{
 int i,j,min,u,s;
 for(i=0;i<n;i++)
    {
  d[i]=cost[src][i];
  v[i]=0;
  p[i]=src;
    }
 v[src]=1;
 for(i=0;i<n;i++)
    {
  min=99;
  u=0;
  for(j=0;j<n;j++)
      {
   if(!v[j])
   if(d[j]<min)
       {
    min=d[j];
    u=j;
       }
      }
  v[u]=1;
  for(s=0;s<n;s++)
  if(!v[s]&&(d[u]+cost[u][s]<d[s]))
```

```
        {
    d[s]=d[u]+cost[u][s];
    p[s]=u;
        }
      }
}
void dj::display(int src)
{
 int i,k,parent;
 for(i=0;i<n;i++)
      {
  if(i==src)
  continue;
  cout<<"The shortest path from "<<src<<" to "<<i<<" is
"<<endl;
  k=i;
  cout<<k<<"<----";
  while(p[k]!=src)
        {
   cout<<p[k]<<"<---";
   k=p[k];
        }
  cout<<src<<endl;
  cout<<"and the distance is "<<d[i]<<endl;
      }
}
int main()
{
 int source;
 dj dij;
 dij.read_matrix();
 cout<<"enter the source"<<endl;
 cin>>source;
 dij.short_path(source);
 dij.display(source);
 return 0;
}
```

2] Write a program for error detecting code using CRC-CCITT (16-bits).

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

```
char data[100], concatdata[117], src_crc[17],
dest_crc[17], frame[120], divident[18], divisor[18] =
"10001000000100001", res[17] = "0000000000000000";

void crc_cal(int node)
{
    int i, j;
    for (j = 17; j <= strlen(concatdata); j++)
    {
        if (divident[0] == '1')
        {
            for (i = 1; i <= 16; i++)
                if (divident[i] != divisor[i])
                    divident[i-1] = '1';
                else
                    divident[i-1] = '0';
        }
        else
        {
            for (i = 1; i <= 16; i++)
                divident[i-1] = divident[i];
        }
        if (node == 0)
            divident[i-1] = concatdata[j];
        else
            divident[i-1] = frame[j];
    }
    divident[i-1] = '\0';
    printf("\ncrc is %s\n", divident);
    if (node == 0)
        strcpy(src_crc, divident);
    else
        strcpy(dest_crc, divident);
}

int main()
{
    int i;
    printf("\n\t\t\tAT SOURCE NODE\n\nenter the data to
be sent : ");
    fgets(data, sizeof(data), stdin);
    data[strcspn(data, "\n")] = '\0'; // remove newline

    strcpy(concatdata, data);
    strcat(concatdata, "0000000000000000");
    for (i = 0; i <= 16; i++)
        divident[i] = concatdata[i];
```

```
    divident[i+1] = '\0';

    crc_cal(0);

    printf("\ndata is :\t");
    puts(data);
    printf("\nthe frame transmitted is :\t");
    printf("\n%s%s", data, src_crc);
    printf("\n\t\tSOURCE NODE TRANSMITTED THE FRAME ----
>");

    printf("\n\n\n\n\t\t\tAT DESTINATION NODE\nenter the
received frame:\t");
    fgets(frame, sizeof(frame), stdin);
    frame[strcspn(frame, "\n")] = '\0'; // remove newline

    for (i = 0; i <= 16; i++)
        divident[i] = frame[i];
    divident[i+1] = '\0';

    crc_cal(1);

    if (strcmp(dest_crc, res) == 0)
        printf("\nreceived frame is error free ");
    else
        printf("\nreceived frame has one or more
errors");

    return 0;
}
```

AT SOURCE NODE

enter the data to be sent : 1101011011

crc is 0001110101011010

data is : 1101011011

the frame transmitted is :

11010110110001110101011010

SOURCE NODE TRANSMITTED THE FRAME ---->

AT DESTINATION NODE

enter the received frame: 1101111011000111010101101 0

crc is 1010101110101011

received frame has one or more errors

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

```
#include<stdio.h>

struct rtable
{
 int dist[20],nextnode[20];
}table[20];

int cost[10][10],n;
void distvector()
{
 int i,j,k,count=0;
 for(i=0;i<n;i++)
 {
  for(j=0;j<n;j++)
  {
   table[i].dist[j]=cost[i][j];
   table[i].nextnode[j]=j;
  }
 }
  do
  {
   count=0;
  for(i=0;i<n;i++)
  {
   for(j=0;j<n;j++)
    {
```

```
    for(k=0;k<n;k++)
     {
      if(table[i].dist[j]>cost[i][k]+table[k].dist[j])
      {
       table[i].dist[j]=table[i].dist[k]+table[k].dist[j];
       table[i].nextnode[j]=k;
       count++;
      }
     }
    }
   }

 }while(count!=0);
}

int main()
{
 int i,j;
 printf("\nenter the no of vertices:\t");
 scanf("%d",&n);
 printf("\nenter the cost matrix\n");
 for(i=0;i<n;i++)
   for(j=0;j<n;j++)
    scanf("%d",&cost[i][j]);
 distvector();
 for(i=0;i<n;i++)
 {
  printf("\nstate value for router %c \n",i+65);
  printf("\ndestnode\tnextnode\tdistance\n");
  for(j=0;j<n;j++)
  {
   if(table[i].dist[j]==99)
    printf("%c\t\t-\t\tinfinite\n",j+65);
   else


printf("%c\t\t%c\t\t%d\n",j+65,table[i].nextnode[j]+65,ta
ble[i].dist[j]);
 }       }
return 0;
 }
```

4] Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.

Server.c

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
void error(char *msg)
{
perror(msg);
exit(1);
}
int main(int argc,char *argv[])
{
int sockfd,newsockfd,portno,clilen,n,i=0;
char buffer[256],c[2000],ch;
struct sockaddr_in serv_addr,cli_addr;
FILE *fd;
if(argc < 2)
{
fprintf(stderr,"ERROR,no port provided\n");
exit(1);
}
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0)
error("ERROR opening socket");
bzero((char*) &serv_addr,sizeof(serv_addr));
portno=atoi(argv[1]);
serv_addr.sin_family=AF_INET;
serv_addr.sin_addr.s_addr=INADDR_ANY;
serv_addr.sin_port=htons(portno);
if(bind(sockfd,(struct
sockaddr*)&serv_addr,sizeof(serv_addr))<0)
error("ERROR on binding");
listen(sockfd,5);
clilen=sizeof(cli_addr);
printf("SERVER:Waiting for client....\n");
newsockfd=accept(sockfd,(struct sockaddr*)
&cli_addr,&clilen);
if(newsockfd<0)
error("ERROR on accept");
bzero(buffer,256);
n=read(newsockfd,buffer,255);
if(n<0)
error("ERROR reading from socket");
```

```
printf("SERVER:%s \n",buffer);
if((fd=freopen(buffer,"r",stdin))!=NULL)
{
printf("SERVER:%s found! \n Transfering the contents
...\n",buffer);
while((ch=getc(stdin))!=EOF)
c[i++]=ch;
c[i]='\0';
printf("File content %s\n",c);
n=write(newsockfd,c,1999);
if(n<0)
error("ERROR in writing to socket");
}
else
{
printf("SERVER:File not found!\n");
n=write(newsockfd,"File not found!",15);
if(n<0)
error("ERROR writing to socket");
}
return 0;
}
```

Client.c

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<string.h>
#include<unistd.h>
#include<stdlib.h>
void error(char *msg)
{
perror(msg);
exit(0);
}
int main(int argc,char *argv[])
{int sockfd,portno,n;
struct sockaddr_in serv_addr;
struct hostent *server;
char filepath[256],buf[3000];
if(argc < 3)
{
```

```
fprintf(stderr,"usage %s hostname port\n",argv[0]);
exit(0);
}
portno=atoi(argv[2]);
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0)
error("\nerror in opening socket");
printf("\nclient online");
server=gethostbyname(argv[1]);
if(server==NULL)
{
fprintf(stderr,"error ,no such host");
exit(0);
}
printf("\n server online");
bzero((struct sockaddr_in
*)&serv_addr,sizeof(serv_addr));
serv_addr.sin_family=AF_INET;
bcopy((char *)server->h_addr,(char
*)&serv_addr.sin_addr.s_addr,server->h_length);
serv_addr.sin_port=htons(portno);
if(connect(sockfd,(struct sockaddr
*)&serv_addr,sizeof(serv_addr))<0)

  error("error writing to socket");
printf("\nclient:enter path with filename:\n");
scanf("%s",filepath);
n=write(sockfd,filepath,strlen(filepath));
if(n<0)
error("\nerror writing to socket");
bzero(buf,3000);
n=read(sockfd,buf,2999);
if(n<0)
error("\nerror reading to socket");
printf("\nclient:displaying from socket");

fputs(buf,stdout);
return 0;
}
```

5] Write a program for error detecting using Hamming Code.

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

```
void genhamcode();
void makeerror();
void correcterror();
int h[12];
int main()
{
int i,ch;
printf("\n enter the message in bits\n");
for(i=1;i<12;i++)
if(i==3||i==5||i==6||i==7||i==9||i==10||i==11)
scanf("%d",&h[i]);
for(i=1;i<12;i++)
printf("%d",h[i]);
genhamcode();
printf("\n do you want to make error\n(0 or 1)\n");
scanf("%d",&ch);
if(ch)
{
makeerror();
correcterror();
}
else
printf("\n no error");
return(0);
}
void genhamcode()
{
int temp,i;
temp=h[3]+h[5]+h[7]+h[9]+h[11];
(temp%2!=0)?(h[1]=1):(h[1]=0);
temp=h[3]+h[6]+h[7]+h[10]+h[11];
(temp%2!=0)?(h[2]=1):(h[2]=0);
temp=h[5]+h[6]+h[7];
(temp%2!=0)?(h[4]=1):(h[4]=0);
temp=h[9]+h[10]+h[11];
(temp%2!=0)?(h[8]=1):(h[8]=0);
printf("\n transmitted codeword is:\n");
for(i=1;i<12;i++)
printf(" %d ",h[i]);}
void makeerror()
{
int pos,i;
printf("\n enter the position you want to make error\n");
scanf("%d",&pos);
if(h[pos]==1)
h[pos]=0;
else
```

```
h[pos]=1;
printf("\n Error occured and the error codeword is\n");
for(i=1;i<12;i++)
printf(" %d ",h[i]);
}
void correcterror()
{
int r1,r2,r4,r8,i,errpos;
r1=(h[1]+h[3]+h[5]+h[7]+h[9]+h[11])%2;
r2=(h[2]+h[3]+h[6]+h[7]+h[10]+h[11])%2;
r4=(h[4]+h[5]+h[6]+h[7])%2;
r8=(h[8]+h[9]+h[10]+h[11])%2;
errpos=r8*8+r4*4+r2*2+r1*1;
printf("\n Error occured in pos %d\n",errpos);
printf("\n\n..... correction starts
now.....\n");
if(h[errpos]==1)
h[errpos]=0;
else
h[errpos]=1;
printf("\n Original codeword is :");
for(i=1;i<12;i++)
printf(" %d ",h[i]);
}
```

6] Write a Program to implement sliding window protocol

Sender.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define MAX 1024

int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;
    char buffer[MAX] = {0};
    int total_frames, window_size;
    int base = 0, next_frame = 0;
    int ack_no;
```

```
    printf("Enter total number of frames to send: ");
    scanf("%d", &total_frames);

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

    // Create socket
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("Socket creation error");
        exit(EXIT_FAILURE);
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);
    serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    // Connect to receiver
    if (connect(sock, (struct sockaddr *)&serv_addr,
sizeof(serv_addr)) < 0) {
        perror("Connection failed");
        exit(EXIT_FAILURE);
    }

    printf("Connected to receiver.\n");

    // Sliding window simulation
    while (base < total_frames) {
        while (next_frame < base + window_size &&
next_frame < total_frames) {
            char msg[10];
            sprintf(msg, "%d", next_frame);
            send(sock, msg, strlen(msg), 0);
            printf("Frame %d sent\n", next_frame);
            next_frame++;
        }

        // Receive ACK
        int valread = read(sock, buffer, MAX);
        buffer[valread] = '\0';
        ack_no = atoi(buffer);
        printf("ACK %d received\n", ack_no);

        if (ack_no > base)
            base = ack_no;
        else
```

```
            printf("ACK out of order or lost, resending
window...\n");

        sleep(1);
    }

    send(sock, "exit", strlen("exit"), 0);
    printf("All frames sent successfully!\n");
    close(sock);
    return 0;
}
```

receiver.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define MAX 1024

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char buffer[MAX] = {0};
    int expected_frame = 0;

    // Create socket
    server_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (server_fd < 0) {
        perror("Socket failed");
        exit(EXIT_FAILURE);
    }

    // Assign address and port
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    // Bind
```

```
    if (bind(server_fd, (struct sockaddr *)&address,
sizeof(address)) < 0) {
        perror("Bind failed");
        exit(EXIT_FAILURE);
    }

    // Listen
    if (listen(server_fd, 3) < 0) {
        perror("Listen");
        exit(EXIT_FAILURE);
    }

    printf("Receiver waiting for connection...\n");
    new_socket = accept(server_fd, (struct sockaddr
*)&address, (socklen_t *)&addrlen);
    if (new_socket < 0) {
        perror("Accept failed");
        exit(EXIT_FAILURE);
    }
    printf("Sender connected!\n");

    // Receive frames
    while (1) {
        int valread = read(new_socket, buffer, MAX);
        if (valread <= 0)
            break;

        buffer[valread] = '\0';
        if (strcmp(buffer, "exit") == 0)
            break;

        int frame_no = atoi(buffer);
        printf("Received Frame: %d\n", frame_no);

        if (frame_no == expected_frame) {
            printf("Frame %d received correctly. Sending
ACK %d\n", frame_no, frame_no + 1);
            expected_frame++;
        } else {
            printf("Frame %d out of order. Expected
%d\n", frame_no, expected_frame);
        }

        char ack[10];
        sprintf(ack, "%d", expected_frame);
        send(new_socket, ack, strlen(ack), 0);
    }
```

```
    printf("Connection closed.\n");
    close(new_socket);
    close(server_fd);
    return 0;
}
```

7] Write a program to implement FIFO-Client and FIFO-Server to transfer files.

Server.c

```
#include<stdio.h>
#include<stdlib.h>
#include<errno.h>
#include<string.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
#define FIFO1_NAME "fifo1"
#define FIFO2_NAME "fifo2"
int main()
{
char p[100],f[100],c[300],ch;
int num,num2,f1,fd,fd2,i=0;
mknod(FIFO1_NAME,S_IFIFO |0666,0);
mknod(FIFO2_NAME,S_IFIFO |0666,0);
printf("\nSERVER ONLINE");
fd=open(FIFO1_NAME,O_RDONLY);
printf("client online\nwaiting for request\n\n");
while(1)
{
if((num=read(fd,p,100))==-1)
perror("\nread error");
else
{
p[num]='\0';
if((f1=open(p,O_RDONLY))<0)
{
printf("\nserver: %s not found",p);
exit(1);
}
else
{
```

```
printf("\nserver:%s found \ntranfering the contents",p);
stdin=fdopen(f1,"r");
while((ch=getc(stdin))!=EOF)c[i++]=ch;
c[i]='\0';
printf("\nfile contents %s\n ",c);
fd2=open(FIFO2_NAME,O_WRONLY);
if(num2=write(fd2,c,strlen(c))==-1)
perror("\ntranfer error");
else
printf("\nserver :tranfer completed");
}
exit(1);
}
}
}
```

Client.c

```
#include<stdio.h>
#include<stdlib.h>
#include<errno.h>
#include<string.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
#define FIFO1_NAME "fifo1"
#define FIFO2_NAME "fifo2"
int main()
{
char p[100],f[100],c[3000];
int num,num2,f1,fd,fd2;
mknod(FIFO1_NAME,S_IFIFO|0666,0);
mknod(FIFO2_NAME,S_IFIFO|0666,0);
printf("\n waiting for server...\n");
fd=open(FIFO1_NAME,O_WRONLY);
printf("\n SERVER ONLINE !\n CLIENT:Enter the path\n");
while(gets(p),!feof(stdin))
{

if((num=write(fd,p,strlen(p)))==-1)
perror("write error\n");
```

```
else
{
printf("Waiting for reply....\n");
fd2=open(FIFO2_NAME,O_RDONLY);
if((num2=read(fd2,c,3000))==-1)
perror("Transfer error!\n");
else
{

printf("File recieved! displaying the contents:\n");
if(fputs(c,stdout)==EOF)
perror("print error\n");
exit(1);
}
}
}
}
```