```
/* M&H-2 (scheme-2) */

# include<iostream.h>
# include<stdio.h>
# include<math.h>
# include<conio.h>
# include<stdlib.h>

//fig. 12.9
# define N 4
# define M 2


main(void)                          /*main function starts*/
{
int i,j,k,x,no,y,p,q,pg,l,count,IND[5],OND[5],ND[5],z1,z,temp,m,cutset[2],TR,OND1[5],IND1[5];
//fig. 12.9
int I[5][3]={2,0,0,6,8,0,1,5,0,7,0,0,4,3,0};
int O[5][3]={1,3,0,7,0,0,4,0,0,5,8,0,2,6,0};


/*for number of inputs*/
for(i=0;i<=N;i++)
{
count=0;
    for (j=0;j<=M;j++)
    {

    if(I[i][j]!=0)
    count++;
    }
    IND1[i]=count;
}

/*for number of outputs*/
for(i=0;i<=N;i++)
{
count=0;
    for (j=0;j<=M;j++)
    {

    if(O[i][j]!=0)
    count++;
    }
    OND1[i]=count;
}

/*calculate weight of input stream*/
for(i=0;i<=N;i++)
{
    IND[i]=IND1[i]*OND1[i];
}


//printing the values of input weight
/*for(i=0;i<=N;i++)
printf("Input weight=%d \t",IND[i]);*/


//calculate the weight of output stream

for(i=0;i<=N;i++)
{
    z=0;
    for (j=0;j<=M;j++)
    {
    count=0;
    TR=0;
    if(O[i][j]==0)  break;
    else
    {
        for(p=0;p<=N;p++)
        {
            for (k=0;k<=M;k++)
            {
                if(O[i][j]==I[p][k])
```

```
                {
                    for (l=0;l<=M;l++)
                    {
                        if(O[p][l]!=0)
                        {
                            count++;
                            TR=1;

                        }

                    }
                    OND1[z]=count;
                    z++;
                }
                if(TR==1) break;
            }
            if(TR==1) break;
        }
        }
        if(j==0)    OND[i]=0;
        OND[i]=OND[i]+OND1[z-1];
    }
}


//printing the values of input weight
/*for(i=0;i<=N;i++)
printf("Output weight=%d \t",OND[i]);*/



//calculation of node denomination(ND)
for(i=0;i<=N;i++)
ND[i]=(OND[i]- IND[i]);

//printing the values of ND
/*for(i=0;i<=N;i++)
printf("ND=%d \t",ND[i]);*/

//check for nodes with zero or negative ND and replace input streams by precursors
for(i=0;i<=N;i++)
{
    count=0;
    if(ND[i]<=0)
    {
        for(p=0;p<=N;p++)
            {
                for (j=0;j<=M;j++)
                {
                    if(I[i][0]==O[p][j])
                    {
                        count=1;
                        temp=I[i][1];
                        for (k=0;k<=M;k++)
                        {
                            I[i][k]=I[p][k];
                            if(I[p][k+1]==0)
                            {
                                I[i][k+1]=temp;
                                m=k+1;
                                break;
                            }
                        }

                    }
                    if(count==1)break;
                }
                if(count==1)break;
            }
        count=0;
        if(I[i][m]!=0)
        {
            for(p=0;p<=N;p++)
            {
            for (j=0;j<=M;j++)
                {
```

```
                if(I[i][m]==O[p][j])
                        {
                            count=1;
                            for (k=0;k<=M;k++)
                            {
                                I[i][m]=I[p][k];
                                if(I[p][k+1]==0)
                                    break;

                            }
                        }
                if(count==1)break;
                }
                if(count==1)break;
            }
        }
    }
}

//printing the values of input and output streams after replacement
/*printf("The values of input and output streams after replacement\n");
for(i=0;i<=N;i++)
{
    for (j=0;j<=M;j++)
    {
    printf("Input=%d \t",I[i][j]);
    printf("Output=%d \n",O[i][j]);
    }
}*/

//To find out the cutset
p=0;
for(i=0;i<=N;i++)
{
for (j=0;j<=M;j++)
    {
    for (j=0;j<=M;j++)
    {
        if(I[i][j]==O[i][k])
        {
            if(I[i][j]!=cutset[p])
            {
            cutset[p]=I[i][j];
            p++;
            }
        }
    }
    }
}

//Removing the cutset streams from the list
for(i=0;i<=N;i++)
{
    for (j=0;j<=M;j++)
    {
        for (k=0;k<=p;k++)
        {
        if(I[i][j]==cutset[k])
            I[i][j]=0;
        if(O[i][j]==cutset[k])
            O[i][j]=0;
        }
    }
}

for(i=0;i<=N;i++)
{
    count=0;
    for (j=0;j<=M;j++)
    {
        if(I[i][j]!=0)
            count++;
    }
    if(count==0)
    for (j=0;j<=M;j++)
        O[i][j]=0;
```

```
}

for(i=0;i<=N;i++)
{
    count=0;
    for (j=0;j<=M;j++)
    {
        if(O[i][j]!=0)
            count++;
    }
    if(count==0)
    for (j=0;j<=M;j++)
        I[i][j]=0;
}

//print the values of list after reduction
/*printf("The values of list after reduction\n");
for(i=0;i<=N;i++)
{
    for (j=0;j<=M;j++)
    {
    printf("Input=%d \t",I[i][j]);
    printf("Output=%d \n",O[i][j]);
    }
}*/

// print the value of cutset
printf("The value of cutset are\n");

for(i=0;i<p;i++)
printf("Cutset=%d \n",cutset[i] );

}       /*main ends*/
```