```
/* M&H-2(scheme-1) */

# include<iostream.h>
# include<stdio.h>
# include<math.h>
# include<conio.h>
# include<stdlib.h>
# define TMAX 15
# define TMAX1 15

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

//fig. 12.10
# define N 18
# define M 14

//fig. 12.11
//# define N 24
//# define M 14

main(void)                          /*main function starts*/
{
int i,j,k,l,x,no,y,p,q,pg,count,p1,IND[N+2],OND[N+2],ND[N+2],z1,z,temp,m,cutset[N],tmax,I1[N+2][1
5],I2[N+2][15],A1[N+2],tmax1;

//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};

//fig. 12.10
int I[19][15]={1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10,0,0,0,0,0,0,0,0,0,0,0,0,0,0,11,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,3,9,0,0,0,0,0,0,0,0,0,0,0,0,8,20,0,0,0,0,0,0,0,0,0,0,0,
0,0,12,13,23,26,31,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,7,18,0,0,0,0,0,0,0,0,0,0,0,0,0,19,0,0,0,0,0,0,0,0,0,0,0,0,0,0,21,0,0,0,0,0,0,0,0,0,0,0,0,0,0,29
,30,0,0,0,0,0,0,0,0,0,0,0,0,0,14,0,0,0,0,0,0,0,0,0,0,0,0,0,0,15,0,0,0,0,0,0,0,0,0,0,0,0,0,0,16,17
,0,0,0,0,0,0,0,0,0,0,0,0,0,22,25,0,0,0,0,0,0,0,0,0,0,0,0,0,24,27,0,0,0,0,0,0,0,0,0,0,0,0,0,28,0,0
,0,0,0,0,0,0,0,0,0,0,0,0};
int O[19][15]={2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,12,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,1,3,4,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,10,11,13,21,0,0,0,0,0,0,0,
0,0,0,28,30,0,0,0,0,0,0,0,0,0,0,0,0,0,14,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,22,23,0,0,0,0,0,0,0,0,0,0,0,0,0,31
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,15,0,0,0,0,0,0,0,0,0,0,0,0,0,0,16,0,0,0,0,0,0,0,0,0,0,0,0,0,0,18,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,17,19,24,0,0,0,0,0,0,0,0,0,0,0,0,25,26,0,0,0,0,0,0,0,0,0,0,0,0,0,27,29,0
,0,0,0,0,0,0,0,0,0,0,0,0};

//fig. 12.11
//int I[25][15]={16,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,9,0,0,0,0,0,0,0,0,0,0,0,0,0,10,11,0,0,0,0,0,0,0,0,0,0,0,0,0,12,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
13,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,17,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,19,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,32,0,0,0,0,0,0,0,0,0,0,0,0,0,0,18,20,0,0,0,0,0,0,0,0,0,0,0,0,0,22,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,14,0,0,0,0,0,0,0,0,0,0,0,0,0,0,21,0,0,0,0,0,0,0,0,0,0,0,0,0,0,23,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,15,24,27,33,0,0,0,0,0,0,0,0,0,0,0,25,0,0,0,0,0,0,0,0,0,0,0,0,0,0,26,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,28,29,0,0,0,0,0,0,0,0,0,0,0,0,0,30,0,0,0,0,0,0,0,0,0,0,0,0,0,0,31,0,0,0,0,0,0,0
,0,0,0,0,0,0,0};
//int O[25][15]={2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,5,0,0,0,0,0,0,0,0,
0,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,9,0,0,0,0,0,0,0,0,0,0,0,0
,0,10,0,0,0,0,0,0,0,0,0,0,0,0,0,12,0,0,0,0,0,0,0,0,0,0,0,0,0,0,13,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
4,15,0,0,0,0,0,0,0,0,0,0,0,0,0,25,0,0,0,0,0,0,0,0,0,0,0,0,0,0,17,0,0,0,0,0,0,0,0,0,0,0,0,0,0,18,2
9,0,0,0,0,0,0,0,0,0,0,0,0,0,19,0,0,0,0,0,0,0,0,0,0,0,0,0,0,21,0,0,0,0,0,0,0,0,0,0,0,0,0,0,11,20,0
,0,0,0,0,0,0,0,0,0,0,0,0,22,0,0,0,0,0,0,0,0,0,0,0,0,0,0,23,0,0,0,0,0,0,0,0,0,0,0,0,0,0,24,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,16,0,0,0,0,0,0,0,0,0,0,0,0,0,0,26,27,0,0,0,0,0,0,0,0,0,0,0,0,0,28,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,30,0,0,0,0,0,0,0,0,0,0,0,0,0,0,31,0,0,0,0,0,0,0,0,0,0,0,0,0,0,32,33,0,0,0,0,0,0
,0,0,0,0,0,0,0};


for(i=0;i<=N;i++)
{
count=0;

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

    IND[i]=count;

}



for(i=0;i<=N;i++)
{
count=0;

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

    OND[i]=count;
}




/*eliminate nodes with single precursors*/

for(tmax1=0;tmax1<=TMAX1;tmax1++)
{
    for(i=0;i<=N;i++)
    {
        count=0;
        for (j=0;j<=M;j++)
        {
        if(IND[i]==1 && OND[i]==1)
        {
        for(l=0;l<=N;l++)
            {
            for (k=0;k<=M;k++)
                {
                    if(I[l][k]==O[i][j])
                    {
                    I[l][k]=I[i][j];
                    count=1;
                    }
                }
            }
        } if(count==1) break;
        }
    }
}


for(i=0;i<=N;i++)
    {
        if(IND[i]==1 && OND[i]==1)
            {
                I[i][0]=0;
                O[i][0]=0;
            }

    }



//calculate weight of input stream
for(i=0;i<=N;i++)
{
count=0;
if(I[i][0]!=0)
{
    for (j=0;j<=M;j++)
    {

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

}


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


//calculate the weight of output stream

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

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


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

//printing the values of ND
/*printf("\n\nPrinting the Node Denomination\n");
for(i=0;i<=N;i++)
printf("ND[%d]=%d \t",i,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 (j=0;j<=M;j++)
        {
            count=0;
            if(I[i][j]>0)
            {
                for(q=0;q<=N;q++)
                {
                    if(O[q][0]>0)
                    {
                        for(l=0;l<=M;l++)
                        {
                            if(O[q][l]>0)
                            {
                                if(I[i][j]==O[q][l])
                                {
                                    if((ND[q]<=0) && q<i)
                                    {
                                        if(j==0)    x=j;
                                        for (k=0;k<=M;k++)
                                        {
                                            I1[i][x]=I1[q][k];
                                            x++;
                                            count=1;
                                            if(I1[q][k+1]==0) break;
                                        }
```

```
                                           }
                                           else
                                           {
                                               if(j==0)     x=j;
                                               for (k=0;k<=M;k++)
                                               {
                                                   I1[i][x]=I[q][k];
                                                   x++;
                                                   count=1;
                                                   if(I[q][k+1]==0) break;
                                               }
                                           }
                                       }
                                   }if(count==1) break;
                               }
                           }if(count==1) break;
                       }
                   }
               }
           }
       }


//printing the values after removing nodes with single input & output
/*printf(" \nprinting the values \n");

for(i=0;i<=N;i++)
{
    if(ND[i]<=0)
    {
        if(I[i][0]!=0)
        {
        printf("Node=%d\n",i+1);
        printf("\n");
        for (j=0;j<=M;j++)
            {
            if(I1[i][j]>=0)
            printf("I=%d \t",I[i][j]);
        }
        printf("\n");
        for (j=0;j<=M;j++)
        {
            if(I1[i][j]>=0)
            printf("O=%d \t",O[i][j]);
        }
        printf("\n\n");
        }
    }
}*/

//printing the values after second reduction
/*printf(" \nprinting the values after after second reduction\n");

for(i=0;i<=N;i++)
{
    if(ND[i]<=0)
    {
        if(I1[i][0]>=0)
        {
        printf("Node=%d\n",i+1);
        printf("\n");
        for (j=0;j<=M;j++)
        {
            if(I1[i][j]>=0)
            printf("I1=%d \t",I1[i][j]);

        }
        printf("\n");
        for (j=0;j<=M;j++)
        {
            if(I1[i][j]>=0)
            printf("O=%d \t",O[i][j]);
        }
            printf("\n\n");
        }
    }
```

```
}*/

//Putting all values in one array
for(i=0;i<=N;i++)
{
    if(I[i][0]!=0)
    {
    for(j=0;j<=M;j++)
    {
    if(ND[i]<=0)
            I2[i][j]=I1[i][j];
    else
        I2[i][j]=I[i][j];

    }
    }
}



/*
//printing the values of input and output after putting in one array
printf("printing the vaues of input and output before cutset")
for(i=0;i<=N;i++)
{
        if(I[i][0]!=0)
        printf("Node=%d\n",i+1);
        printf("\n");
        for (j=0;j<=M;j++)
        {
            if(I2[i][j]>0)
            printf("I2=%d \t",I2[i][j]);
            else break;
        }
        printf("\n");
        for (j=0;j<=M;j++)
        {
            if(O[i][j]>0)
            printf("O=%d \t",O[i][j]);
            else break;
        }
            printf("\n");
}
*/

//To find out the cutset
   p=0;

for(i=0;i<=N;i++)
{
    if(I[i][0]!=0)
    {
    for (j=0;j<=M;j++)
    {
    count=0;
    for (k=0;k<=M;k++)
    {
        if(I2[i][j]==O[i][k])
        {
            if(O[i][k]>0)
            {
                cutset[p]=O[i][k];
                p++;
                count=1;
            }

        }if(count==1) break;

    }
    }
    }
}


q=0;
for(i=0;i<p;i++)
```

```
    {
    for(j=0;j<p;j++)
    {
        if(cutset[i]==cutset[j] && i!=j)
        {
            q++;
            for(k=i;k<p;k++)
            {
            cutset[k]=cutset[k+1];

            if(k==p) cutset[k]=0;
            }
        }
    }
}

p1=p-q;
p=p1;

/*printf("The values of cutset");
for(i=0;i<p;i++)
{
    printf(" cutset=%d",cutset[i]);
}*/


/*for(i=0;i<N;i++)
{
    for(j=0;j<M;j++)
    {
    if(O[i][j]!=0)
    {
        printf("The value of node=%d",O[i][j]);
        getch();
    }
    }
}*/




//deleting the input of cutset node by previous input values in the output list
for(i=0;i<=N;i++)
{
    if(O[i][0]!=0)
    {
    for(j=0;j<=M;j++)
    {
        count=0;
        for(k=0;k<p;k++)
        {
        if(O[i][j]==cutset[k] && ND[i]<=0)
        {

            for(l=0;l<=N;l++)
            {
            for(m=0;m<=M;m++)
            {
                for(q=0;q<=M;q++)
                {
                if(O[l][m]==I[i][q])
                {
                    for(z=m;z<=M;z++)
                    {
                    O[l][z]=O[l][z+1];
                        count=1;
                        if(O[l][z+1]==0) break;
                    }
                }
                }
            }
            }
        }if(count==1) break;
        }
    }
    }
}




//deleting the input of cutset node by previous input values in the input list
for(i=0;i<=N;i++)
{
    if(I2[i][0]!=0)
    {
    for(j=0;j<=M;j++)
    {
        count=0;
        for(k=0;k<p;k++)
        {
        if(O[i][j]==cutset[k] && ND[i]<=0)
        {

            for(l=0;l<=N;l++)
            {
            if(I2[l][0]!=0)
            {
            for(m=0;m<=M;m++)
            {
                for(q=0;q<=M;q++)
                {
                if(I2[l][m]==I[i][q])
                {
                    for(z=m;z<=M;z++)
                    {
                    I2[l][z]=I2[l][z+1];
                        count=1;
                        if(I2[l][z+1]==0) break;
                    }
                }
                }
            }
            }
            }
        }if(count==1) break;
        }
    }
    }
}


//Deleting the cutset  from input list
for(i=0;i<=N;i++)
{
 if(I2[i][0]!=0)
    {
    for (j=0;j<=M;j++)
    {
    if(I2[i][j]!=0)
        {
        for (l=0;l<p;l++)
        {
        if(I2[i][j]!=0)
        {
         if(I2[i][j]==cutset[l])
         {

             for (k=j;k<=M;k++)
            {
                I2[i][k]=I2[i][k+1];
                 if(I2[i][k+1]==0) break;
            }
             }
        }
        }
        }
    }
    }
}
```

```
//Deleting the cutset  from output list
for(i=0;i<=N;i++)
{
 if(O[i][0]!=0)
    {
    for (j=0;j<=M;j++)
    {
    if(O[i][j]!=0)
        {
        for (l=0;l<p;l++)
        {
         if(O[i][j]==cutset[l])
         {

             for (k=j;k<=M;k++)
            {
                 O[i][k]=O[i][k+1];
                  if(O[i][k+1]==0) break;
            }

         }
        }
        }
    }
    }
}

//preventing the repetation
for(i=0;i<=N;i++)
{
    for(j=0;j<=M;j++)
    {
        if(I2[i][j]==I2[i][j+1])
        {
            for(k=j;k<=M;k++)
            {
            I2[i][k]=I2[i][k+1];
            if(I2[i][k+1]==0) break;
            }
        }
    }
}

//removing the input if there are no output
for(i=0;i<=N;i++)
{
    for(j=0;j<=M;j++)
    {
        if(O[i][0]==0) I2[i][j]=0;
    }
}

//Innovation
x=0;
for(i=0;i<=N;i++)
{
    for(j=0;j<=M;j++)
    {
        count=0;
        if(O[i][j]>0)
        {
            for(k=0;k<=N;k++)
            {
                for(l=0;l<=M;l++)
                {
                    if(I2[k][l]>0)
                    {
                        if(O[i][j]==I2[k][l])
                        {

                            count=1;
                        }if(count==1) break;
                    }if(count==1) break;
                }
            }
        }
        if(count==0)
                {
                A1[x]=O[i][j];
                x++;
                }
        if(O[i][j+1]==0) break;
    }
}




//printing the values after the innovations
/*for(i=0;i<x;i++)
{
printf("A1  %d=%d\n",i,A1[i]);
getch();
}*/

//If the node is not present in the input then remove it
for(i=0;i<=N;i++)
{
    for(j=0;j<=M;j++)
    {
        for(k=0;k<x;k++)
        {
        if(O[i][j]==A1[k])
        {
            for(l=j;l<=M;l++)
            {
            O[i][l]=O[i][l+1];
            if(O[i][l+1]==0)
            {
                for(m=l+1;m<=M;m++)
                    O[i][m]=0;
                break;
            }
            }
        }
        }
    }
}


for(i=0;i<=N;i++)
{
        for (j=0;j<=M;j++)
        {
            if(I2[i][j]>0)
            {
                if(I2[i][j+1]==0)
                {
                    for(k=j+1;k<=M;k++)
                    {
                        I2[i][k]=0;
                    }
                }
            }
        }
}

for(i=0;i<=N;i++)
{
        for (j=0;j<=M;j++)
        {
            if(O[i][j]>0)
            {
                if(O[i][j+1]==0)
                {
                    for(k=j+1;k<=M;k++)
                    {
                        O[i][k]=0;
                    }
                }
            }
        }
}


//removing the input if there are no output
for(i=0;i<=N;i++)
{
    for(j=0;j<=M;j++)
    {
        if(O[i][0]==0) I2[i][j]=0;
    }
}

//removing the output if there are no input
for(i=0;i<=N;i++)
{
    for(j=0;j<=M;j++)
    {
        if(I2[i][0]==0) O[i][j]=0;
    }
}


for(i=0;i<=N;i++)
{
count=0;

    for (j=0;j<=M;j++)
    {
    if(I2[i][j]>0)
    count++;
    }

    IND[i]=count;

}

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


for(i=0;i<=N;i++)
{
count=0;

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

    OND[i]=count;
}


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


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

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

```
/*eliminate nodes with single precursors*/

for(tmax1=0;tmax1<=TMAX1;tmax1++)
{
    for(i=0;i<=N;i++)
    {
        if(I2[i][j]>0 && O[i][j]>0 )
        {
        count=0;
        for (j=0;j<=M;j++)
        {
        if(ND[i]==0)
        {
        for(l=0;l<=N;l++)
            {
            for (k=0;k<=M;k++)
                {
                    if(I2[l][k]==O[i][j])
                    {
                    I2[l][k]=I2[i][j];
                    count=1;
                    }
                }
            }
        } if(count==1) break;
        }
        }
    }
}


for(i=0;i<=N;i++)
    {
        if(ND[i]==0)
            {
                I2[i][0]=0;
                O[i][0]=0;
            }

    }

//To find out the cutset
for(i=0;i<=N;i++)
{
    if(I2[i][0]>0)
    {
    for (j=0;j<=M;j++)
    {
    count=0;
    for (k=0;k<=M;k++)
    {
        if(I2[i][j]==O[i][k])
        {
            if(O[i][k]>0)
            {
                cutset[p]=O[i][k];
                p++;
                count=1;
            }

        }if(count==1) break;

    }
    }
    }
}

q=0;
for(i=0;i<p;i++)
    {
    for(j=0;j<p;j++)
    {
        if(cutset[i]==cutset[j] && i!=j)
        {
            q++;
            for(k=i;k<p;k++)
            {
            cutset[k]=cutset[k+1];

            if(k==p) cutset[k]=0;
            }
        }
    }
}

p=p-q;

/*printf("\nThe values of input and output values after deleting cutset and input of cuset\n");
for(i=0;i<=N;i++)
{
        if(I2[i][0]>0)
        {
        printf("Node=%d\n",i);
        printf("\n");
        }
        for (j=0;j<=M;j++)
        {
            if(I2[i][j]>0)
            {
            printf("I=%d \t",I2[i][j]);
            getch();
            }
            else break;
        }
        printf("\n");
        for (j=0;j<=M;j++)
        {
            if(O[i][j]>0)
            {
            printf("O=%d \t",O[i][j]);
            getch();
            }
            else break;
        }
            printf("\n");
}*/

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


}/*main*/
```