```
/* This is the generalized code for B & M Alogorithm*/

# include<iostream.h>
# include<stdio.h>
# include<math.h>
# include<conio.h>
# include<stdlib.h>
# define TMAX 10


//Fig.12.9
//# define N 7  //no of nodes
//# define P 1  //no of precursors

//Fig.12.10
//# define N 30
//# define P 4

//Fig.12.11
# define N 31
# define P 4



main(void)                          /*main function starts*/
{
int i,a,b,j,k,p,h,q,pg,count,tmax,w,x,y,z,c1[42],c2[42],TR,l,cutset[10],count1,count2,found,imp[3
0],check,done;


//fig. 12.9
//int no[8]={1,2,3,4,5,6,7,8};
//int pr[8][2]={2,0,3,4,2,0,1,5,7,0,3,5,6,8,7,0};

//fig 12.10
//int no[31]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
};
//int pr[31][5]={5,0,0,0,0,1,0,0,0,0,5,0,0,0,0,5,0,0,0,0,6,0,0,0,0,7,18,0,0,0,8,20,0,0,0,2,3,9,0,
0,10,0,0,0,0,8,20,0,0,0,8,20,0,0,0,11,0,0,0,0,8,20,0,0,0,4,0,0,0,0,14,0,0,0,0,15,0,0,0,0,22,25,0,
0,0,16,17,0,0,0,22,25,0,0,0,19,0,0,0,0,8,20,0,0,0,21,0,0,0,0,21,0,0,0,0,22,25,0,0,0,24,27,0,0,0,2
4,27,0,0,0,28,0,0,0,0,12,13,23,26,31,28,0,0,0,0,12,13,23,26,31,29,30,0,0,0};

//fig 12.11
int no[32]={2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,
33};
int pr[32][5]={16,0,0,0,0,2,0,0,0,0,3,0,0,0,0,3,0,0,0,0,4,0,0,0,0,6,0,0,0,0,7,0,0,0,0,7,0,0,0,0,9
,0,0,0,0,22,0,0,0,0,10,11,0,0,0,12,0,0,0,0,13,0,0,0,0,13,0,0,0,0,15,24,27,33,0,8,0,0,0,0,19,0,0,0
,0,32,0,0,0,0,22,0,0,0,0,18,20,0,0,0,14,0,0,0,0,21,0,0,0,0,23,0,0,0,0,5,17,0,0,0,24,27,0,0,0,25,0
,0,0,0,26,0,0,0,0,19,0,0,0,0,28,29,0,0,0,30,0,0,0,0,31,0,0,0,0};


//printing the values
/*for(i=0;i<=N;i++)
{
    printf("node=%d     ",no[i] );
    for (j=0;j<=P;j++)
    printf("pre =%d",pr[i][j]);
    printf("\n");
    getch();

}*/

/*eliminate nodes with single precursors*/

for(tmax=0;tmax<=TMAX;tmax++)
{
    for(i=0;i<=N;i++)
    {
        count=0;
        for (j=0;j<=P;j++)
        {
        if(pr[i][j]!=0)
        count++;
        }
//printf("count=%d\n",count);
    if(count==1)
```

```
        {
        for(p=0;p<=N;p++)
            {
            for (k=0;k<=P;k++)
                {
                if(pr[p][k]==no[i])
                pr[p][k]=pr[i][0];
                }
            }
        no[i]=0;
        pr[i][0]=0;
        }/*if*/
    }/*for*/
}

for(i=0;i<=N;i++)
{
    if(no[i]!=0)
    {
    for(j=0;j<=P;j++)
    {
        found=0;
        for(k=j+1;k<=P;k++)
        {
            if(pr[i][j]==pr[i][k])
            {
                if(pr[i][k]!=0)
                {
                    for(l=0;l<=P;l++)
                pr[i][l]=pr[i][l+1];
                }
            }
        }
    }
    }
}



printf("\n\n\n");

//printing the values after first reduction
 /*printf("The values after first reduction\n");

for(i=0;i<=N;i++)
{
    printf("node=%d     ",no[i] );
    for (j=0;j<=P;j++)
    printf("pre =%d",pr[i][j]);
    printf("\n");
    getch();

}*/

//cutset
p=0;
for(i=0;i<=N;i++)
{
    if(no[i]!=0)
    {
for (j=0;j<=P;j++)
    {

        if(pr[i][j]==no[i])
        {
        cutset[p]=no[i];
        p++;
        }
    }
    }
}



for(i=0;i<=N;i++)
{
    if(no[i]!=0)
    {
    for (j=0;j<=P;j++)
    {
        for (k=0;k<=P;k++)
        {
            if(no[i]==cutset[k])
            {
                for(l=0;l<=P;l++)
                {
                    pr[i][l]=0;
                }
            }
        }
    }
    }
}

/* print the value of cutset*/
/*for(i=0;i<p;i++)
printf("Cutset=%d \n",cutset[i] );*/
//checkig the pair
z=p;
y=0;
for(i=0;i<=N;i++)
{
    TR=0;
    if(no[i]!=0)
    {
        for (j=0;j<=P;j++)
        {
            for (l=0;l<=N;l++)
            {
                found=0;

                    if(pr[i][j]==no[l])
                    {
                    found=1;
                    for (k=0;k<=P;k++)
                        {

                        if(pr[l][k]==no[i])
                        {
                            done=0;
                            count1=0;
                            check=0;
                            for(a=0;a<=P;a++)
                            {
                                if(pr[i][a]!=0)
                                count1++;

                            }
                            c1[i]=count1-1;
                            count2=0;
                            for(h=0;h<=P;h++)
                            {
                                if(pr[l][h]!=0)
                                count2++;
                            }
                            c2[l]=count2-1;
                            //printf("i=%d\n",i+1);
                            //printf("l=%d\n",l+1);
                            //printf("C1=%d\n",c1[i]);
                            //printf("C2=%d\n",c2[l]);


                            if(c1[i]>=c2[l])
                            {
                                imp[y]=no[l];
                                y++;
                                //printf("the value of imp[%d]=%d",y-1,imp[y-1]);
                                //printf("the value of node[i]%d",no[i]);
                                //printf("the value of node[l]=%d\n",no[l]);
                                for(x=0;x<y;x++)
                                {
                                    if(no[i]==imp[x])
                                    {
                                        //printf("the value of node [i] in if =%d",no[i]);
                                        check=1;
                                    }

                                }
                                if(check!=1)
                                {
                                    for(b=0;b<z;b++)
                                    {
                                        if(no[i]==cutset[b])
                                        {
                                            //printf("the value of node [i] in if 1 =%d",no[i]);
                                            done=1;
                                        }
                                    }
                                    if(done!=1)
                                    {
                                    cutset[z]=no[i];
                                    z++;
                                    TR=1;
                                    }
                                }
                            }
                            else
                            {
                                //printf("I am in else");
                                imp[y]=no[i];
                                y++;
                                check=0;
                                done=0;
                                for(x=0;x<y;x++)
                                {
                                if(no[i]==imp[x])
                                    check=1;
                                }
                                if(check!=1)
                                {
                                    for(b=0;b<z;b++)
                                    {
                                        if(no[i]==cutset[b])
                                            done=1;
                                    }
                                    if(done!=1)
                                    {
                                    cutset[z]=no[l];
                                    z++;
                                    TR=1;
                                    }
                                }
                            }
                            //cutset[z]=no[i];
                            //z++;
                            //TR=1;
                            //if(z!=0)
                        //printf("Cutset %d=%d \n\n",z-1,cutset[z-1]);

                        }
                        }if(TR==1) break;
                    }
                    if(found==1) break;
                   if(TR==1) break;
                }  if(TR==1) break;
            }
        }
}

b=z-1;
/*printf("\n");
printf("total number of cutset=%d\n",z);*/

// print the value of cutset
for(a=0;a<=b;a++)
{
printf("Cutset %d=%d \n",a,cutset[a] );
}
```

```
}
```