```
/* Generalised Code for M & H-1 Algorithm*/

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

#define maxnode 19  /*Maximum node */
#define maxinp 6  /*Maximum input for any node*/
#define maxout 5  /*Maximum output for any node*/

main()
{
    int b,c,d,e,i,j,k,l,m,n,p,r,s,w,x,y,z;
    int no_sing_i,no_sing_o,no_sing_io,no_self,temp,count_out,count_in,count_out_sd,count_sd1,ccc
ount;
    int noinp[maxnode],noout[maxnode];
    int inp[maxnode][maxinp], out[maxnode][maxout],cutset[maxnode],sd[maxnode];

    printf("\n\n\tThis is a generalised code for M and H-1 tearing algorithm\n\n");
    printf("\tEnter the input stream for each node in order\n");

    // initilisation of the input stream of all the nodes
    for(i=0;i<maxnode;i++)
    {
        for(j=0;j<maxinp;j++)
            inp[i][j]=0;
    }

    for(i=0;i<maxnode;i++)
    {
        printf("\nEnter the number of input stream for node %d\t",i+1);
        scanf("%d",&noinp[i]);
        for(j=0;j<noinp[i];j++)
        {
            printf("\nEnter the input stream \t");
            scanf(" %d",&inp[i][j]);
        }
        printf("\n\n");
    }

    // initilisation of the output stream of all the nodes
    for(i=0;i<maxnode;i++)
    {
        for(j=0;j<maxout;j++)
            out[i][j]=0;
    }

    for(i=0;i<maxnode;i++)
    {
        printf("\nEnter the number of output stream for node %d\t",i+1);
        scanf("%d",&noout[i]);
        for(j=0;j<noout[i];j++)
        {
            printf("\nEnter the output stream \t");
            scanf(" %d",&out[i][j]);
        }
        printf("\n\n");
    }

    // initialisation of the cutset and Succesive digit
    for(i=0;i<maxnode;i++)
    {
        cutset[i]=0;
        sd[i]=0;
    }

    //nodes with single input-output streams and replace single output in input list by precursor
    for(i=0;i<maxnode;i++)
    {
        if(noinp[i]==1 && noout[i]==1)
        {
            for(j=0;j<maxnode;j++)
            {
                for(k=0;k<noinp[j];k++)
                {
                    if(inp[j][k]==out[i][0])
```

```
                        inp[j][k]=inp[i][0];
                }
            }
            inp[i][0]=0;
        }
    }

mark3:

    //sorting the values in the descending order for the inputs and outputs of all the nodes
    for(l=0;l<maxnode;l++)
    {
        for(r=0;r<(maxinp-1);r++)
        {
            for(s=r+1;s<maxinp;s++)
            {
                if(inp[l][r]<inp[l][s])
                {
                    c=inp[l][r];
                    b=inp[l][s];
                    inp[l][r]=b;
                    inp[l][s]=c;
                }
            }
        }

        for(r=0;r<(maxout-1);r++)
        {
            for(s=r+1;s<maxout;s++)
            {
                if(out[l][r]<out[l][s])
                {
                    d=out[l][r];
                    e=out[l][s];
                    out[l][r]=d;
                    out[l][s]=e;
                }
            }
        }
    }

    //nodes with single input-output streams and replace single output in input list by precursor

    no_sing_io=0;
    for(i=0;i<maxnode;i++)
    {
        no_sing_i=no_sing_o=0;
        for(j=0;j<maxinp;j++)
        {
            if(inp[i][j]!=0)
                no_sing_i++;
        }

        for(j=0;j<maxout;j++)
        {
            if(out[i][j]!=0)
                no_sing_o++;
        }

        if(no_sing_i==1 && no_sing_o==1)
        {
            no_sing_io++;
            for(j=0;j<maxnode;j++)
            {
                for(k=0;k<maxinp;k++)
                {
                    if(inp[j][k]!=0)
                    {
                        if(inp[j][k]==out[i][0])
                        {
                            inp[j][k]=inp[i][0];
                            inp[i][0]=0;
                            goto mark01;
                        }
                    }
                }
```

```
            }
        }
mark01:
        printf("");
    }

    if(no_sing_io==0)
        goto mark2;

    printf("\n\n ");

    /* displaying the input streams after reduction*/

    printf("\n \t\tInput after reduction  ");
    for(i=0;i<maxnode;i++)
    {
        printf("\n \t For node %d ",i+1);
        for(j=0;j<noinp[i];j++)
            printf("  %d",inp[i][j]);
    }
    printf("\n\n ");
    getch();

mark4:

    /* finding the self loop*/
    no_self=0;
    for(i=0;i<maxnode;i++)
    {
        for(j=0;j<maxinp;j++)
        {
            if(inp[i][j]!=0)
            {
                for(k=0;k<noout[i];k++)
                {
                    if(inp[i][j]==out[i][k])
                    {
                        no_self++;
                        for(l=0;l<maxnode;l++)
                        {
                            if(cutset[l]==0)
                            {
                                cutset[l]=inp[i][j];
                                inp[i][j]=out[i][k]=0;
                                goto mark1;
                            }
                        }
                    }
                }
            }
            mark1:
            printf("");
        }
    }

    if(no_self!=0)
        goto mark3;

mark2:
    printf("\n\n");

    //sorting the values in the descending order for the inputs and outputs of all the nodes
    for(l=0;l<maxnode;l++)
    {
        for(r=0;r<(maxinp-1);r++)
        {
            for(s=r+1;s<maxinp;s++)
            {
                if(inp[l][r]<inp[l][s])
                {
                    c=inp[l][r];
                    b=inp[l][s];
                    inp[l][r]=b;
                    inp[l][s]=c;
                }
            }
```

```
    }

    for(r=0;r<(maxout-1);r++)
    {
        for(s=r+1;s<maxout;s++)
        {
            if(out[l][r]<out[l][s])
            {
                d=out[l][r];
                e=out[l][s];
                out[l][r]=d;
                out[l][s]=e;
            }
        }
    }
}

// finding the successive digit for all the nodes
for(m=0;m<maxnode;m++)
{
    count_out=0;
    if(inp[m][0]!=0)
    {
        for(n=0;n<noout[m];n++)
        {
            if(out[m][n]!=0)
            {
                temp=out[m][n];
                count_out++;
            }
        }

        if(count_out==1)
        {
            for(l=0;l<maxnode;l++)
            {
                for(n=0;n<noinp[l];n++)
                {
                    if(inp[l][n]!=0)
                    {
                        if(temp==inp[l][n])
                        {
                            count_out_sd=0;
                            for(p=0;p<noout[l];p++)
                            {
                                if(out[l][p]!=0)
                                    count_out_sd++;
                            }
                            sd[m]=count_out_sd;
                        }
                    }
                }
            }
        }
    }
}

// printing the Successive digit value for all the nodes
printf(" \n \n The Succesive digit for the nodes are listed below\n\n");
for(i=0;i<maxnode;i++)
{
    printf(" \n For node %d \t %d",i+1,sd[i]);
    if(i==11)
        getch();
}

//sorting the values in the descending order for the inputs and outputs of all the nodes
for(l=0;l<maxnode;l++)
{
    for(r=0;r<(maxinp-1);r++)
    {
        for(s=r+1;s<maxinp;s++)
        {
            if(inp[l][r]<inp[l][s])
            {
                c=inp[l][r];
```

```
                b=inp[l][s];
                inp[l][r]=b;
                inp[l][s]=c;
            }
        }
    }

    for(r=0;r<(maxout-1);r++)
    {
        for(s=r+1;s<maxout;s++)
        {
            if(out[l][r]<out[l][s])
            {
                d=out[l][r];
                e=out[l][s];
                out[l][r]=d;
                out[l][s]=e;
            }
        }
    }
}

count_sd1=0;
for(i=0;i<maxnode;i++)
{
    if(sd[i]==1)
    {
        count_sd1++;
        for(j=0;j<maxnode;j++)
        {
            for(n=0;n<maxinp;n++)
            {
                if(inp[j][n]!=0)
                {
                    if(inp[j][n]==out[i][0])
                    {
                        inp[j][n]=0;
                        for(k=0;k<maxinp;k++)
                        {
                            if(inp[i][k]!=0)
                            {
                                for(l=0;l<maxinp;l++)
                                {
                                    if(inp[j][l]==0)
                                    {
                                        inp[j][l]=inp[i][k];
                                        inp[i][k]=0;
                                        break;
                                    }
                                }
                            }
                        }
                        goto mark5;
                    }
                }
            }
        }
    }
    mark5:
    printf("");
}

//sorting the values in the descending order for the inputs and outputs of all the nodes
for(l=0;l<maxnode;l++)
{
    for(r=0;r<(maxinp-1);r++)
    {
        for(s=r+1;s<maxinp;s++)
        {
            if(inp[l][r]<inp[l][s])
            {
                c=inp[l][r];
                b=inp[l][s];
                inp[l][r]=b;
                inp[l][s]=c;
            }
```

```
            }
        }

        for(r=0;r<(maxout-1);r++)
        {
            for(s=r+1;s<maxout;s++)
            {
                if(out[l][r]<out[l][s])
                {
                    d=out[l][r];
                    e=out[l][s];
                    out[l][r]=d;
                    out[l][s]=e;
                }
            }
        }
    }

    if(count_sd1!=0)
        goto mark4;
    printf(" \n");

    /* displaying the input streams after reduction*/

    printf("\n \t\tInput after reduction \n\n ");
    for(i=0;i<maxnode;i++)
    {
        printf("\n \t For node %d ",i+1);
        for(j=0;j<noinp[i];j++)
            printf("  %d",inp[i][j]);
    }
    printf("\n\n ");
    getch();

/* problem specific code */

    for(i=0;i<maxnode;i++)
    {
        if(sd[i]>=2)
        {
            if(sd[i]==2)
            {
                for(j=0;j<maxnode;j++)
                {
                    int ccount=0;
                    for(n=0;n<maxinp;n++)
                    {
                        if(inp[j][n]!=0)
                        {
                            if(inp[j][n]==out[i][0])
                            {
                                z=n;
                                for(y=0;y<maxinp;y++)
                                {
                                    if(z!=y)
                                    {
                                        if(inp[j][y]!=0)
                                            ccount++;
                                    }
                                }
                                if(ccount==0)
                                {
                                    for(x=0;x<maxinp;x++)
                                    {
                                        for(y=0;y<maxnode;y++)
                                        {
                                            for(w=0;w<maxout;w++)
                                            {
                                                if(inp[i][x]==out[y][w])
                                                {
                                                    inp[i][x]=out[y][w]=0;
                                                    goto mark7;
                                                }
                                            }
                                        }
mark7:
```

```
                                    printf("");
                                }
                                for(x=0;x<maxout;x++)
                                {
                                    for(y=0;y<maxnode;y++)
                                    {
                                        for(w=0;w<maxinp;w++)
                                        {
                                            if(out[j][x]==inp[y][w])
                                            {
                                                out[j][x]=inp[y][w]=0;
                                                goto mark8;
                                            }
                                        }
                                    }
mark8:
                                    printf("");
                                }
                            }
                            for(l=0;l<maxnode;l++)
                            {
                                if(cutset[l]==0)
                                {
                                    cutset[l]=inp[j][n];
                                    inp[j][n]=out[i][0]=0;
                                    goto mark9;
                                }
                            }
                        }
                    }
                }
            }
mark9:
            if(sd[i]==3)
            {
                for(j=0;j<maxnode;j++)
                {
                    int ccount=0;
                    for(n=0;n<maxinp;n++)
                    {
                        if(inp[j][n]!=0)
                        {
                            if(inp[j][n]==out[i][0])
                            {
                                z=n;
                                for(y=0;y<maxinp;y++)
                                {
                                    if(z!=y)
                                    {
                                        if(inp[j][y]!=0)
                                            ccount++;
                                    }
                                }
                                if(ccount==0)
                                {
                                    for(x=0;x<maxinp;x++)
                                    {
                                        for(y=0;y<maxnode;y++)
                                        {
                                            for(w=0;w<maxout;w++)
                                            {
                                                if(inp[i][x]==out[y][w])
                                                {
                                                    inp[i][x]=out[y][w]=0;
                                                    goto mark91;
                                                }
                                            }
                                        }
mark91:
                                        printf("");
                                    }
                                    for(x=0;x<maxout;x++)
                                    {
                                        for(y=0;y<maxnode;y++)
                                        {
```

```
                                        for(w=0;w<maxinp;w++)
                                        {
                                            if(out[j][x]==inp[y][w])
                                            {
                                                out[j][x]=inp[y][w]=0;
                                                goto mark92;
                                            }
                                        }
                                    }
mark92:
                                    printf("");
                                }
                            }
                            for(l=0;l<maxnode;l++)
                            {
                                if(cutset[l]==0)
                                {
                                    cutset[l]=inp[j][n];
                                    inp[j][n]=out[i][0]=0;
                                    goto mark93;
                                }
                            }
                        }
                    }
                }
            }
mark93:
        printf("");
        }
    }

    //sorting the values in the descending order for the inputs and outputs of all the nodes
    for(l=0;l<maxnode;l++)
    {
        for(r=0;r<(maxinp-1);r++)
        {
            for(s=r+1;s<maxinp;s++)
            {
                if(inp[l][r]<inp[l][s])
                {
                    c=inp[l][r];
                    b=inp[l][s];
                    inp[l][r]=b;
                    inp[l][s]=c;
                }
            }
        }

        for(r=0;r<(maxout-1);r++)
        {
            for(s=r+1;s<maxout;s++)
            {
                if(out[l][r]<out[l][s])
                {
                    d=out[l][r];
                    e=out[l][s];
                    out[l][r]=d;
                    out[l][s]=e;
                }
            }
        }
    }

    goto mark3;

    for(i=0;i<maxnode;i++)
    {
        if(sd[i]==2)
        {
            for(j=0;j<maxnode;j++)
            {
                for(n=0;n<maxinp;n++)
                {
                    if(inp[j][n]!=0)
                    {
```

```
                    if(inp[j][n]==out[i][0])
                    {
                        inp[j][n]=0;
                        for(k=0;k<maxinp;k++)
                        {
                            if(inp[i][k]!=0)
                            {
                                for(l=0;l<maxinp;l++)
                                {
                                    if(inp[j][l]==0)
                                    {
                                        inp[j][l]=inp[i][k];
                                        inp[i][k]=0;
                                        break;
                                    }
                                }
                            }
                        }
                        goto mark6;
                    }
                }
            }
        }
    }
    mark6:
    printf("*");
}

//sorting the values in the descending order for the inputs and outputs of all the nodes
for(l=0;l<maxnode;l++)
{
    for(r=0;r<(maxinp-1);r++)
    {
        for(s=r+1;s<maxinp;s++)
        {
            if(inp[l][r]<inp[l][s])
            {
                c=inp[l][r];
                b=inp[l][s];
                inp[l][r]=b;
                inp[l][s]=c;
            }
        }
    }

    for(r=0;r<(maxout-1);r++)
    {
        for(s=r+1;s<maxout;s++)
        {
            if(out[l][r]<out[l][s])
            {
                d=out[l][r];
                e=out[l][s];
                out[l][r]=d;
                out[l][s]=e;
            }
        }
    }
}

goto mark4;

cccount=0;
for(i=0;i<maxnode;i++)
{
    if(inp[i][0]!=0)
        cccount++;
}

printf("\n\n The following are the cutset of the given process\n\n");
for(i=0;i<maxnode;i++)
{
    if(cutset[i]!=0)
        printf("\t %d",cutset[i]);
}
```

```
}
```