#include <stdio.h>

#include <stdlib.h>

#define STACK\_INIT\_SIZE 100

#define STACKINCREMENT 10

#define MAX 20

typedef int VertexType;

typedef struct ArcNode//表结点

{

int adjvex;//弧所指向的顶点的位置

struct ArcNode \*nextarc;

}ArcNode;

typedef struct VNode//头结点

{

VertexType data;//顶点信息

ArcNode \*firstarc;//指向第一条依附该弧的顶点指针

}VNode,\*AdjList;

typedef struct

{

AdjList vertices;

int vexnum;//图的\*\*当前\*\*顶点数

}ALGraph;

typedef struct//栈的定义

{

int \*base;

int \*top;

int stacksize;

}SqStack;

/////////栈的操作函数定义

void initialStack(SqStack \*s)

{

s->base=(int \*)malloc(STACK\_INIT\_SIZE\*sizeof(int));

if(!s->base) exit(0);

s->top=s->base;

s->stacksize=STACK\_INIT\_SIZE;

}

void Push(SqStack \*s,int e)

{

if(s->top-s->base>=s->stacksize)

{

s->base=(int \*)realloc(s->base,(STACK\_INIT\_SIZE+STACKINCREMENT)\*sizeof(int));

if(!s->base) exit(0);

s->top=s->base+s->stacksize;

s->stacksize+=STACKINCREMENT;

}

\*(s->top)++=e;

}

void Pop(SqStack \*s,int \*e)

{

if(s->top==s->base) exit(0);

\*e=\*--(s->top);

}

void GetTop(SqStack \*s,int \*e)

{

if(s->top==s->base) exit(0);

\*e=\*(s->top-1);

}

int StackEmpty(SqStack \*s)

{

if(s->base==s->top)

return(1);

else

return(0);

}

/////创建图的邻接矩阵

void CreatAjacentMatrix(int \*array,int n)//创建邻接矩矩阵(n行n列)

{

int a;

int i,j,flag=0;

printf("请输入一个%d行%d列的关于图的邻接矩阵:\n",n,n);

for(i=0;i<n;i++)

for(j=0;j<n;j++)

{

scanf("%d",&a);

\*(array+i\*n+j)=a;

}

}

void PrintAjacentMatrix(int \*array,int n)

{

int i,j;

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

printf("%5d ",\*(array+i\*n+j));

printf("\n");

}

}

////将邻接矩阵导出为图的邻接表形式

void CreatAdjList(int \*array,int n,ALGraph \*G)

{

int i,j;

ArcNode \*p;//表结点

G->vexnum=n;//初始化顶点数

G->vertices=(VNode \*)malloc((n+1)\*sizeof(VNode));//头结点数组,开辟n+1长度的数组空间

for(i=1;i<=n;i++)//初始化头结点数组

{

G->vertices[i].data=i;

G->vertices[i].firstarc=NULL;

}

//////////

for(i=0;i<n;i++)

for(j=0;j<n;j++)

{

if(\*(array+i\*n+j)==1)

{

p=(ArcNode \*)malloc(sizeof(ArcNode));

p->adjvex=j+1;

p->nextarc=G->vertices[i+1].firstarc;

G->vertices[i+1].firstarc=p;

}

}

}

void FindInDegree(ALGraph G,int \*indegree)//对顶点求入度

{

int i,j;

ArcNode \*p;

for(i=1;i<=G.vexnum;i++)

indegree[i]=0;//indispensable

for(i=1;i<=G.vexnum;i++)//对每个结点跑完整个邻接表

for(j=1;j<=G.vexnum;j++)

for(p=G.vertices[j].firstarc;p;p=p->nextarc)

if(G.vertices[i].data==p->adjvex)//==

indegree[i]++;

}

/////////拓扑排序算法

int TopologicalSort(ALGraph G)

{

//有向图采用邻接表存储结构

//若G无回路，则flag=0,输出G的顶点的一个拓扑序列,否则给出该有向图有回路的提示.

int i,count,k;

int \*indegree=(int \*)malloc((G.vexnum+1)\*sizeof(int));

SqStack S;

ArcNode \*p;

FindInDegree(G,indegree);//对顶点求入度indegree[G.vexnum]

initialStack(&S);//为避免重复检测入度为0的顶点，可另设一栈暂存放所有入度为0的顶点

for(i=1;i<=G.vexnum;i++)

if(!indegree[i])

Push(&S,i);//0入度点进栈

count=0;//对输出顶点计数，作为判断是否有回路的根据

while(!StackEmpty(&S))

{

Pop(&S,&i);

printf("%d ",i);//输出i号顶点并计数

count++;

for(p=G.vertices[i].firstarc;p;p=p->nextarc)

{

k=p->adjvex;//表结点的数据域，即对i号顶点的每个邻接点的入度减1

if(!(--indegree[k]))//若入度减少为0，则入栈

Push(&S,k);

}

}

if(count<G.vexnum)//该有向图有回路

return 0;

else

return 1;

}

void main()

{

int n;

int \*A;

ALGraph G;

printf("请输入你想创建的邻接矩矩阵的行列数(即顶点数):\n");

scanf("%d",&n);

A=(int \*)malloc(n\*n\*sizeof(int));

CreatAjacentMatrix(A,n);

printf("请输出图的邻接矩阵A:\n");

PrintAjacentMatrix(A,n);

CreatAdjList(A,n,&G);

printf("该有向图的一个拓扑排序结果如下所示:\n");

if(TopologicalSort(G))

printf("\n");

else

printf("该有向图有回路!\n");

}