#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <stdbool.h>

#include <malloc.h>

void

delmz(double\*\* a, double n)

{

double min = 0.0001;

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

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

if(fabs(a[i][j]) < min){

a[i][j] = 0;

}

}

}

}

//ввод(1) или инициализация нулями(0) матрицы nxn

double\*\*

in(int n, bool flag)

{

double\*\* mas = (double\*\*)malloc(n\*sizeof(double\*));

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

mas[i] = (double\*)malloc(n\*sizeof(double));

}

if(flag){

printf("Ввведите матрицу %dx%d: \n",n,n);

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

printf("%d: ",i);

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

scanf("%lf",&mas[i][j]);

}

}

}else{

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

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

mas[i][j] = 0;

}

}

}

return mas;

}

//инициализация единичной матрицы nxn

double\*\*

init(int n)

{

double\*\* E = (double\*\*)malloc(n\*sizeof(double\*));

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

E[i] = (double\*)malloc(n\*sizeof(double));

}

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

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

E[i][j] = 0;

}

E[i][i] = 1;

}

return E;

}

//ввод(1) или инициализация нулями(0) столбца nx1

double\*

inb(int n, bool flag)

{

double\* cur = (double\*)malloc(n\*sizeof(double));

if(flag){

printf("Введите столбец решений b\n");

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

scanf("%lf",&cur[i]);

}

}else{

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

cur[i] = 0;

}

}

return cur;

}

//освобождение памяти из под матрицы mas nxn

void

Fr(double\*\* mas, int n)

{

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

{

free(mas[i]);

}

free(mas);

}

//форматированный вывод матрицы mas nxn

void

printSLAU(double\*\* mas,int n)

{

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

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

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

printf("% -.3lf ",mas[i][j]);

}

printf(" | \n\n");

}

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

}

//форматированный вывод матрицы mas nxn и столбца решений b nx1

void

printSLAUF(double\*\* mas,double\* b,int n)

{

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

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

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

printf("% -.3lf ",mas[i][j]);

}

printf(" | % -.3lf \n",b[i]);

}

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

}

//копирование из матрицы from nxn в матрицу to nxn

void

copy(double\*\* to, double\*\* from, int n)

{

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

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

to[i][j] = from[i][j];

}

}

}

//копирование столбца from nx1 в столбец to nx1

void

copyb(double\* to, double\* from, int n)

{

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

to[i] = from[i];

}

}

//помянять местами строки под номерами from и to в матрице mas nxn

void

swap(double\*\* mas,int from, int to, int n)

{

double\* cur = inb(n,0);

copyb(cur,mas[from],n);

copyb(mas[from],mas[to],n);

copyb(mas[to],cur,n);

free(cur);

}

//деление строки на число

void

divide(double\*\* A, int num, double count, int n)

{

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

A[num][i]/=count;

}

}

//отнять от строки cur строку base умноженную на число count из матрицы A

void

minus(double\*\* A, int base, int cur, double count, int n)

{

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

A[cur][i] = A[cur][i] - A[base][i]\*count;

}

}

//вернуть 1 если double - строка nx1 - нулевая

bool

strdz(double\* a, int n)

{

int count = 0;

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

if(a[i] == 0){

count++;

}

}

return (count != n);

}

//вернуть 1 если int - строка nx1 - единичная

bool

striz(int\* a, int n)

{

int count = 0;

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

if(a[i] == 1){

count++;

}

}

return (count != n);

}

//сортирует по убыванию

void

sort(double\*\* A, int n, int count)

{

int a[count];

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

bool flag = false;

for(int j = 0; j < n && !flag; j++){

if(A[i][j] != 0){

a[i] = j;

flag = true;

}

}

}

for(int i = 0; i < count-1; i++){

for(int j = i+1; j < count; j++){

if(a[j]<a[i]){

int tmp = a[j];

a[j] = a[i];

a[i] = tmp;

swap(A,i,j,n);

}

}

}

}

//сортирует вместе со строкой решений

void

sortF(double\*\* B, double\* b, int n)

{

int a[n];

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

bool flag = false;

for(int j = 0; j < n && !flag; j++){

if(B[i][j] != 0){

a[i] = j;

flag = true;

}

}

if(!flag){

a[i] = n;

}

}

for(int i = 0; i < n-1; i++){

for(int j = i+1; j < n; j++){

if(a[j]<a[i]){

int tmp = a[j];

a[j] = a[i];

a[i] = tmp;

double tmpd = b[i];

b[i] = b[j];

b[j] = tmpd;

swap(B,i,j,n);

}

}

}

}

//сведение матрицы в трапецевидную форму - свап нулевых строк на ненулевые

int

RangAndToTrapezoid(double\*\* A,int n)

{

double min = 0.0000001;

int count = 0;

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

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

if(strdz(A[i],n)){

ind[i] = 1;

count++;

}else{

ind[i] = 0;

}

}

while(striz(ind,count)){

int from = -1;

int to = -1;

int i = 0;

while(ind[i]){

i++;

}

from = i;

while(!ind[i]){

i++;

}

to = i;

if(from < n && to < n){

swap(A,from,to,n);

ind[from] = 1;

ind[to] = 0;

}

}

free(ind);

sort(A,n,count);

return count;

}

//ищёт первый столбец, такой что на count месте стоит не 0

int

findfst(double\*\* A, int n, int count)

{

bool flag = false;

bool flag1 = false;

int i,j;

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

for( j = n-1; j >=0 && !flag; j--){

if(A[j][i] != 0 && j == count){

flag = true;

}

}

if(flag){

return i;

}

}

return -1;

}

//проверка на нормальность локального флага

bool

isnormal(double\* a, bool\* mask, int current, int n)

{

bool flag = true;

for(int i = 0; i < current && flag; i++){

if(mask[i] && a[i]){

flag = false;

}

}

for(int i = current + 1; i < n && flag; i++){

if(mask[i] && a[i]){

flag = false;

}

}

return flag;

}

//проверка на нормальность глобального флага

bool

isallnormal(bool\* a, int n)

{

bool flag = true;

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

flag\*= a[i];

}

return flag;

}

//перевставка, для выражение главных через свободные

void

repaste(double\*\* A, int to, int from, int n)

{

double k = A[to][from];

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

A[to][i] = A[to][i] + k\*A[from][i];

}

A[to][from] = 0;

}

//всё на месте - сортировка до трапецевидной

void

allinplace(double\*\* B, bool\* cur, int n, double\* b)

{

bool\* real = (bool\*)malloc(n\*sizeof(bool));;

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

real[i] = strdz(B[i],n);

}

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

for(int j = 0; cur[i] && !real[i] && j < n; j++)

if(!cur[j] && real[j]){

double tmp = b[i];

b[i] = b[j];

b[j] = tmp;

swap(B,i,j,n);

real[i] = 1;

real[j] = 0;

}

}

free(real);

}

//вывод ФСР

void

printFSR(double\*\* ans, int rang, int n)

{

double min = 0.0001;

for(int i = 0; i < rang; i++){

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

if(fabs(ans[i][j]) < min){

ans[i][j] = 0;

}

}

}

double\* ch = inb(n,0);

rang++;

for(int i = 0; i < n/2; i++){

for(int j = 0; j < rang; j++){

printf(" | %- -.3lf | ",ans[j][i]);

}

printf("\n");

}

for(int j = 0; j < rang-1; j++){

printf(" C%d| %- -.3lf | +",j+1,ans[j][n/2]);

}

printf(" | %- -.3lf | ",ans[rang-1][n/2]);

printf("\n");

for(int i = n/2+1; i < n; i++){

for(int j = 0; j < rang; j++){

printf(" | %- -.3lf | ",ans[j][i]);

}

printf("\n");

}

printf("Некоторое частное решение:\n(");

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

for(int j = 0; j < rang; j++){

ch[i]+=ans[j][i];

}

printf("%- -.3lf ",ch[i]);

}

printf(")\n");

free(ch);

}

//для дальнейшего использования

double\*\*

forout(double\*\* A, bool\* notcur, int rang, double\*b, int n, double\* nor)

{

rang++;

double\*\* ans = (double\*\*)malloc(rang\*sizeof(double\*));

for(int i = 0; i < rang; i++){

ans[i] = (double\*)malloc(n\*sizeof(double));

}

rang--;

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

int k = 0;

int num = 0;

while(k < n){

if(notcur[k]){

ans[num][i] = A[i][k];

num++;

}

k++;

}

ans[rang][i] = b[i];

}

if(rang == n){

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

ans[n][i] = nor[i];

}

}

return ans;

}

void

Frret(double\*\* mas, int rang, int n){

rang++;

for(int i = 0; i < rang; i++){

free(mas[i]);

}

free(mas);

}

//Поиск ФСР

int

fsr(double\*\* A, int n, int rang, double\* b, double\* nor)

{

double\*\* B = in(n,0);

bool allnormal;

bool\* normal = (bool\*)malloc(n\*sizeof(bool));

bool\* cur = (bool\*)malloc(n\*sizeof(bool));

bool\* notcur = (bool\*)malloc(n\*sizeof(bool));

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

normal[i]= 0;

cur[i] = 0;

notcur[i] = 0;

}

for(int i = 0; i < rang; i++){

int eg = findfst(A,n,i);

if(eg != -1){

cur[eg] = 1;

}

}

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

notcur[i] = !cur[i];

}

allinplace(A,cur,n,b);

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

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

B[i][j] = -A[i][j];

}

B[i][i] = 1;

normal[i] = isnormal(B[i],cur,i,n);

}

allnormal = isallnormal(normal,n);

int i = n -1;

bool superflag = 1;

while(!allnormal && superflag){

while(normal[i] && i >=0){

i--;

}

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

if(normal[j] && B[i][j] && i != j && cur[j]){

b[i] = b[i] + b[j]\*B[i][j];

repaste(B,i,j,n);

}

}

normal[i] = isnormal(B[i],cur,i,n);

allnormal = isallnormal(normal,n);

int k = 0;

int chekstr = 1;

int chekstb = 1;

while(cur[k]){

k++;

}

for(int i = 0; i < k;i++){

if(B[k][i] == 0){

chekstr++;

}

if(B[i][k] == 0){

chekstb++;

}

}

for(int i = k+1; i < n;i++){

if(B[k][i] == 0){

chekstr++;

}

if(B[i][k] == 0){

chekstb++;

}

}

if(chekstb == chekstr && chekstr == n){

superflag = false;

}

}

double\*\* forret = forout(B,notcur,n - rang,b,n,nor);

printFSR(forret, n - rang, n);

Fr(B,n);

Frret(forret,n - rang,n);

free(normal);

free(cur);

free(notcur);

}

//приведение матрицы в верхне треугольную, определение детерминанта

//если det! = 0 - нахождение обратной матрицы

//иначе - ФСР

void

gauss(double\*\* A, double\*\* B,double\* b, int n, double\* nor)

{

double min = 0.0001;

double tmp;

double det = 1.0;

for(int i = 0; i < n; i++){//сравнение double и 0

if(A[i][i] == 0){

for(int j = i+1; j < n; j++){

if(A[j][i] != 0){//сравнение double и 0

swap(B,i,j,n);

swap(A,i,j,n);

tmp = b[i];

b[i] = b[j];

b[j] = tmp;

}

}

}

if(fabs(A[i][i]) < min) {

det = 0;

}

det\*=A[i][i];

int k = i;

while(!A[i][k] && k < n){

k++;

}

delmz(A,n);

if(fabs(A[i][k]) > min){

b[i]/=A[i][k];

divide(B,i,A[i][k],n);

divide(A,i,A[i][k],n);

for(int j = i+1; j < n; j++){

b[j] = b[j] - b[i]\*A[j][k];

minus(B,i,j,A[j][k],n);

minus(A,i,j,A[j][k],n);

}

}

}

delmz(A,n);

if(det != 0){

for(int i = n-1; i > 0; i--){

for(int j = i-1; j >= 0; j--){

b[j] = b[j] - b[i]\*A[j][i];

minus(B,i,j,A[j][i],n);

minus(A,i,j,A[j][i],n);

}

}

printf("Определитель равен : % -.3lf\n",det);

printf("Обратная матрица: \n");

printSLAU(B,n);

for(int i = 0; i < n/2; i++){

printf(" | %- -.3lf | \n",b[i]);

}

printf("X =| %- -.3lf | \n",b[n/2]);

for(int i = n/2+1; i < n; i++){

printf(" | %- -.3lf | \n",b[i]);

}

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

}

if(det == 0){

int rang = RangAndToTrapezoid(A,n);

printf("Матрица вырожденна\n");

printf("Её ранг равен: %d\n",rang);

printf("Решение в виде ФСР, Ci -произвольные констаннты,\n");

fsr(A,n,rang,b,nor);

}

}

main(void)

{

int n;

printf("Введите размер квадратной матрицы :\n");

scanf("%d",&n);

double\*\* A = in(n,1);//Ввод матрицы nxn

double\*\* B = in(n,0);//Вспомогательная матрица nxn

double\* a = inb(n,1);//Заполняем столбец решений

double\* b = inb(n,0);//Вспомогательный столбец

double\*\* E = init(n);//инициализация единичной матрицы nxn

sortF(B,b,n);

delmz(B,n);

copy(B,A,n);

copyb(b,a,n);

gauss(B,E,b,n,a);

Fr(A,n);//освобождение памяти

Fr(B,n);

Fr(E,n);

free(a);

free(b);

return 0;

}