**HỌ VÀ TÊN: HUỲNH HỮU HOÀNG**

**LỚP: 22T\_DT1**

**MSSV: 102220063**

#include <iostream>

#include <Eigen/Dense>

#include <vector>

using namespace std;

double a[100][100];

double P[100][100];

double D[100][100];

double P\_Inverse[100][100];

void nhanMaTran(double p[][100], double q[][100], int m, int l, int n, double res[][100]){

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

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

res[i][j] = 0;

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

res[i][j] += p[i][k] \* q[k][j];

}

}

}

}

//mang con

void subMatrix(double a[100][100], int n, int x, int y, double submatrix[100][100]){

// double \*\*submatrix = new double\*[n - 1];

// double submatrix[MAX][MAX];

int subi = 0;

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

int subj = 0;

if(i == y) continue;

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

if(j == x) continue;

submatrix[subi][subj] = a[i][j];

subj++;

}

subi++;

}

}

//

////tinh det

//double determinant(double a[100][100], int n){

// int det = 0;

// if(n == 2){

// return a[0][0] \* a[1][1] - a[1][0] \* a[0][1];

// }

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

// double submatrix[100][100];

// subMatrix(a, n, x, 0, submatrix);

// det += ((x % 2 == 0 ? 1 : - 1) \* a[0][x] \* determinant(submatrix, n - 1));

// }

// return det;

//}

double determinant(double a[][100], int n) {

double det = 1.0;

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

int pivot = i;

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

if (abs(a[j][i]) > abs(a[pivot][i])) {

pivot = j;

}

}

if (pivot != i) {

swap(a[i], a[pivot]);

det \*= -1;

}

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

return 0;

}

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

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

double factor = a[j][i] / a[i][i];

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

a[j][k] -= factor \* a[i][k];

}

}

}

return det;

}

int main() {

int n; cin >> n;

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

for(int j = 0 ; j < n ; j++) cin >> a[i][j];

}

Eigen::MatrixXd A(n, n);

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

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

// std::cout << "A(" << i << ", " << j << "): ";

// std::cin >> A(i, j);

A(i, j) = a[i][j];

}

}

Eigen::EigenSolver<Eigen::MatrixXd> es(A);

std::vector<double> eigenvalues;

for (int i = 0; i < es.eigenvalues().size(); ++i) {

eigenvalues.push\_back(abs(es.eigenvalues()[i].real()) < 1e-4 ? 0 : es.eigenvalues()[i].real());

// eigenvalues.push\_back(es.eigenvalues()[i].real());

}

std::vector<std::vector<double>> eigenvectors;

for (int i = 0; i < es.eigenvectors().cols(); ++i) {

std::vector<double> vec;

for (int j = 0; j < es.eigenvectors().rows(); ++j) {

vec.push\_back(abs(es.eigenvectors()(j, i).real()) < 1e-4 ? 0 : es.eigenvectors()(j, i).real());

// vec.push\_back(es.eigenvectors()(j, i).real());

}

eigenvectors.push\_back(vec);

}

// for (int i = 0; i < eigenvalues.size() - 1; i++){

// int max\_idx = i;

// for (int j = i + 1; j < eigenvalues.size(); j++){

// if(eigenvalues[j] > eigenvalues[max\_idx]) max\_idx = j;

// }

// double tmp = eigenvalues[i];

// eigenvalues[i] = eigenvalues[max\_idx];

// eigenvalues[max\_idx] = tmp;

//

// for(int k = 0 ; k < eigenvectors[i].size(); k++){

// double temp = eigenvectors[i][k];

// eigenvectors[i][k] = eigenvectors[max\_idx][k];

// eigenvectors[max\_idx][k] = temp;

// }

// }

// In eigenvalues

// std::cout << "\nEigenvalues:\n";

// for (const auto& val : eigenvalues) {

// std::cout << val << " ";

// }

for(int i = 0 ; i < eigenvalues.size() ; i++){

D[i][i] = (eigenvalues[i]);

}

// In eigenvectors

// std::cout << "Eigenvectors:\n";

// for (const auto& vec : eigenvectors) {

// for (const auto& val : vec) {

// std::cout << val << " ";

// }

// std::cout << std::endl;

// }

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

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

// V\_Transpose[i][j] = eigenvectors[i][j];

// }

// }

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

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

P[i][j] = eigenvectors[j][i];

}

}

cout << "P:\n";

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

for(int j = 0 ; j < n ; j++) cout << P[i][j] << " ";

cout << endl;

}

cout << "D:\n";

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

for(int j = 0 ; j < n ; j++) cout << D[i][j] << " ";

cout << endl;

}

cout << "P-1:\n";

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

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

double submatrix[100][100];

subMatrix(P, n, j, i, submatrix);

double det = determinant(submatrix, n - 1);

P\_Inverse[i][j] = pow(-1, i + j) \* det;

}

}

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

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

// cout << P\_Inverse[i][j] << " ";

// }

// cout << endl;

// }

double det = determinant(P, n);

if(det == 0){

cout << "Khong the nghich dao ma tran";

return 0;

}

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

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

P\_Inverse[j][i] /= det;

cout << P\_Inverse[j][i] << " ";

}

cout << endl;

}

return 0;

}

