#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <getopt.h>

#include <string.h>

#include <pthread.h>

#include <assert.h>

#include <unistd.h>

#include <time.h>

#define maxN 16384

#define numThreads 2

int kDone = 0;

int N=1;

int globK = 0;

double A[maxN][maxN];

double \*b;

double \*y;

double \*x;

pthread\_mutex\_t mutex1;

pthread\_barrier\_t barrier1;

pthread\_barrier\_t barrier2;

typedef struct{

int i;

int k;

int j;

}parg;

void print\_matrix(double c[maxN][maxN])//print a NxN matrix

//int dim;

{

int i,j;

printf("The %d \* %d matrix is\n", N,N);

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

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

printf("%lf ", c[i][j]);

printf("\n");

}

}

void print\_vector(double c[maxN])//print a N size vector

//int dim;

{

int i,j;

printf("The %d vector is\n",N);

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

printf("%lf ", c[i]);

printf("\n");

}

}

int str\_to\_int( char str\_score[] ) {//method provided by a previous instructor

int value = 0;

for ( int j = 0; str\_score[j] != '\0'; ++j ) {

if ( isdigit( str\_score[j] ) ) {

value = value \* 10 + (str\_score[j] - '0');

} else {

if(str\_score[j]=='.'){//it was a double, just stop.

return value;

}

// was not a number, return -1 to show this

return -1;

}

}

return value;

}

void backSub(){//back substitution

for(int k=N-1;k>=0;k--){

x[k] = y[k];

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

y[i] = y[i]-x[k]\*A[i][k];

}

}

}

void GE\_serial(){//serial gaussian elimination

printf("serial\n");

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

for(int j=k+1;j<N;j++){

A[k][j] = A[k][j]/A[k][k];

}

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

A[k][k]=1;

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

for(int j=k+1;j<N;j++){

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

}

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

A[i][k] = 0;

}

}

}

void \*par3\_1(void \*dat){//thread func

parg \*data = (parg \*)dat;

int i=data->i;

int k=data->k;

while(i<N){

for(int j=k+1;j<N;j++){

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

}

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

A[i][k] = 0;

i=i+numThreads;

}

}

void GE\_par3(){

printf("parrellel3\n");

parg nums[numThreads];

pthread\_t threads[numThreads];

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

for(int j=k+1;j<N;j++){

A[k][j] = A[k][j]/A[k][k];

}

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

A[k][k]=1;

for(int t=0;t<numThreads;t++){

nums[t].k=k;

nums[t].i=k+t+1;

pthread\_create(&threads[t], NULL, par3\_1, &(nums[t]));

}

for(int t=0;t<numThreads;t++){

pthread\_join(threads[t], NULL);

}

}

}

//the main program that interprets command line arguments and runs threads

//

//@param argc number of args

//@Param argv args

//@return failure or success

int main(int argc,char \*argv[]){

char input[10];

scanf("%s", input);

N = str\_to\_int(input);

b = malloc(sizeof(double) \* N);

x = malloc(sizeof(double) \* N);

y = malloc(sizeof(double) \* N);

for(int j=0;j<N;j++){//fill A

for(int i=0;i<N;i++){//2nd row onward, starts A[a][b] b past 0,0

scanf("%s", input);

A[i][j] = atof(input);

}

}

for(int i=0;i<N;i++){//fill b

scanf("%s", input);

b[i] = atof(input);

}

clock\_t start, end;

start = clock();

GE\_par3();

backSub();

print\_vector(x);//print result X

end = clock();

printf(" time elapsed: %f \n", (double)(end - start)/CLOCKS\_PER\_SEC);//print run time of algorithm

free(b);

free(x);

free(y);

}