#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<omp.h>

void main() {

int n;

printf("Enter the dimension of square matrices : ");

scanf\_s("%d", &n);

int i = 0, j = 0, k = 0;

int\*\* arr1 = (int\*)malloc(n \* sizeof(int));

int\*\* arr2 = (int\*)malloc(n \* sizeof(int));

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

omp\_set\_num\_threads(64);

#pragma omp parallel private(j)

{

#pragma omp for

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

srand(i);

arr1[i] = (int\*)malloc(n \* sizeof(int));

arr2[i] = (int\*)malloc(n \* sizeof(int));

res[i] = (int\*)malloc(n \* sizeof(int));

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

arr1[i][j] = rand() % 100;

arr2[i][j] = rand() % 100;

}

}

}

time\_t st, et;

st = clock();

#pragma omp parallel private(j,k)

{

#pragma omp for

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

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

res[i][j] = 0;

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

res[i][j] += arr1[i][k] \* arr2[k][j];

}

}

}

et = clock();

printf("Time taken by parallel algorithm : %lf\n", (double)(et - st) / CLOCKS\_PER\_SEC);

st = clock();

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

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

res[i][j] = 0;

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

res[i][j] += arr1[i][k] \* arr2[k][j];

}

}

et = clock();

printf("Time taken by Sequential algorithm : %lf\n", (double)(et - st) / CLOCKS\_PER\_SEC);

}