**LAB3: MATRIX MULTIPLICATION**

**PRATEEK MOHANTY**

**20BCE1482**

* **MATRIX MULTIPLICATION**

**CODE**

#include <iostream>

using namespace std;

#include <vector>

void read\_matrix(vector<vector<int>> &mat, int n)

{

    int i, j, ele;

    for (i = 0; i < n; i++)

    {

        vector<int> row;

        for (j = 0; j < n; j++)

        {

            cin >> ele;

            row.push\_back(ele);

        }

        mat.push\_back(row);

    }

}

void print\_matrix(vector<vector<int>> mat)

{

    int n = mat.size(), i, j;

    for (i = 0; i < n; i++)

    {

        for (j = 0; j < n; j++)

            cout << mat[i][j] << " ";

        cout << "\n";

    }

}

void square\_matrix\_multiply\_recursive(vector<vector<int>> &mat1,

                                      vector<vector<int>> &mat2, vector<vector<int>> &mat3, int top1, int left1, int top2, int left2, int size)

{

    if (size == 1)

    {

        mat3[top1][left2] += mat1[top1][left1] \* mat2[top2][left2];

    }

    else

    {

        // 11 - top, left

        // 12 - top, left+n/2

        // 22 - top+n/2, left+n/2

        square\_matrix\_multiply\_recursive(mat1, mat2, mat3, top1,

                                         left1, top2, left2, size / 2);

        square\_matrix\_multiply\_recursive(mat1, mat2, mat3, top1, left1 + size / 2, top2 + size / 2, left2, size / 2);

        square\_matrix\_multiply\_recursive(mat1, mat2, mat3, top1,

                                         left1, top2, left2 + size / 2, size / 2);

        square\_matrix\_multiply\_recursive(mat1, mat2, mat3, top1, left1 + size / 2, top2 + size / 2, left2 + size / 2, size / 2);

        square\_matrix\_multiply\_recursive(mat1, mat2, mat3, top1 + size / 2, left1, top2, left2, size / 2);

        square\_matrix\_multiply\_recursive(mat1, mat2, mat3, top1 + size / 2, left1 + size / 2, top2 + size / 2, left2, size / 2);

        square\_matrix\_multiply\_recursive(mat1, mat2, mat3, top1 + size / 2, left1, top2, left2 + size / 2, size / 2);

        square\_matrix\_multiply\_recursive(mat1, mat2, mat3, top1 + size / 2, left1 + size / 2, top2 + size / 2, left2 + size / 2, size / 2);

    }

}

int main()

{

    vector<vector<int>> mat1, mat2;

    int i, j, n;

    cin >> n;

    read\_matrix(mat1, n);

    read\_matrix(mat2, n);

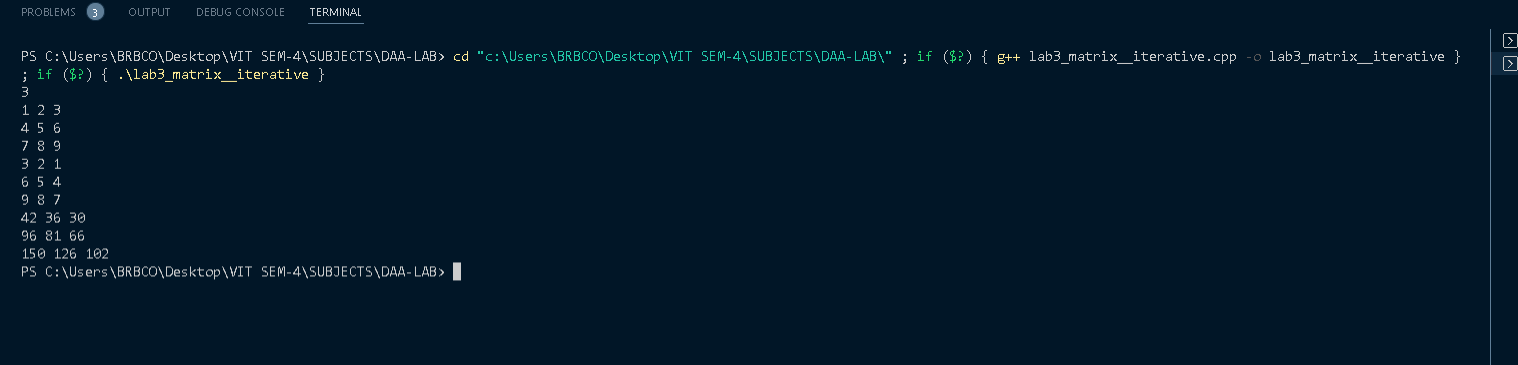
    vector<vector<int>> mat3(n, vector<int>(n, 0));

    square\_matrix\_multiply\_recursive(mat1, mat2, mat3, 0, 0, 0, 0, n);

    print\_matrix(mat3);

}

**OUTPUT**

****

* **ITERATIVE METHOD**

#include <iostream>

using namespace std;

#include <vector>

void read\_matrix(vector<vector<int>> &mat, int n)

{

    int ele;

    for (int i = 0; i < n; i++)

    {

        vector<int> row;

        for (int j = 0; j < n; j++)

        {

            cin >> ele;

            row.push\_back(ele);

        }

        mat.push\_back(row);

    }

}

void print\_matrix(vector<vector<int>> mat)

{

    int n = mat.size(), i, j;

    for (i = 0; i < n; i++)

    {

        for (j = 0; j < n; j++)

            cout << mat[i][j] << " ";

        cout << "\n";

    }

}

void matrix\_multiply(vector<vector<int>> &mat1, vector<vector<int>> &mat2,

                     vector<vector<int>> &mat3)

{

    int i, j, k, n;

    n = mat1.size();

    for (i = 0; i < n; i++)

    {

        for (j = 0; j < n; j++)

        {

            mat3[i][j] = 0;

            for (k = 0; k < n; k++)

            {

                mat3[i][j] += mat1[i][k] \* mat2[k][j];

            }

        }

    }

}

int main()

{

    vector<vector<int>> mat1, mat2;

    int i, j, n;

    cin >> n;

    read\_matrix(mat1, n);

    read\_matrix(mat2, n);

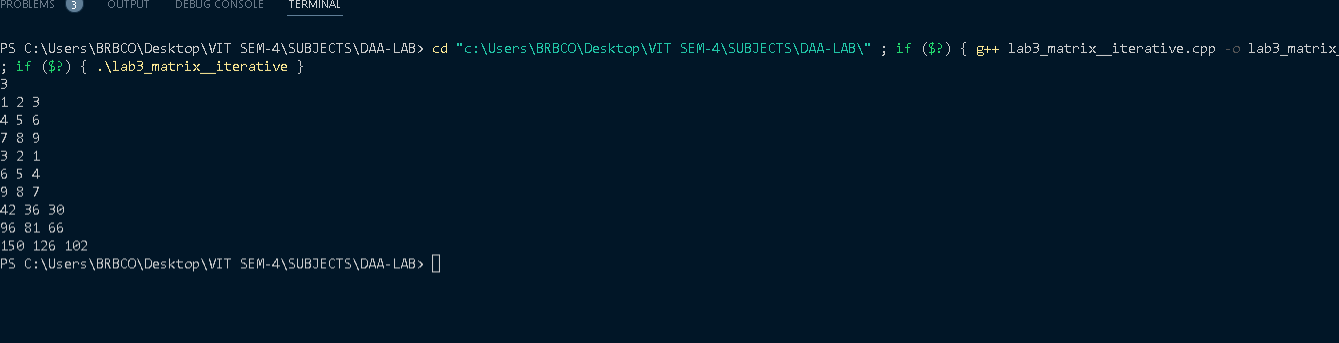
    vector<vector<int>> mat3(n, vector<int>(n, 0));

    matrix\_multiply(mat1, mat2, mat3);

    print\_matrix(mat3);

}

**OUTPUT**

****