Assignment No. 5

Name – Manish Namdev Barage

PRN – 22520007

Batch – T7

Problem Statement - To implement Strassen's Matrix Multiplication algorithm

```
#include <iostream>
#include <vector>
using namespace std;

#define ROW_1 4
#define COL_1 4
#define ROW_2 4
#define COL_2 4
void print(vector<vector<int>> matrix)
{
   for (int i = 0; i < matrix.size(); i++)
   {
      for (int j = 0; j < matrix[i].size(); j++)
      {
         cout << matrix[i][j] << ' ';
      }
      cout << endl;
   }
}
vector<vector<int>> add(vector<vector<int>> A, vector<vector<int>> B, int split_index, int multiplier =
1)
{
   for (auto i = 0; i < split_index; i++)
      for (auto j = 0; j < split_index; j++)
         A[i][j] = A[i][j] + (multiplier * B[i][j]);
```

```
    return A;
}
vector<vector<int>> strassen_multiplication(vector<vector<int>> A, vector<vector<int>> B)
{
    int col_1 = A[0].size();
    int row_1 = A.size();
    int col_2 = B[0].size();
    int row_2 = B.size();

    if (col_1 != row_2)
    {
        cout << "The two matrices cannot be multiplied.";
        return {};
    }

    vector<int> result_row(col_2, 0);
    vector<vector<int>> result(row_1, result_row);

    if (col_1 == 1)
        result[0][0] = A[0][0] * B[0][0];
    else
    {
        int split_index = col_1 / 2;
        vector<int> row_vector(split_index, 0);
        vector<vector<int>> a00(split_index, row_vector);
        vector<vector<int>> a01(split_index, row_vector);
        vector<vector<int>> a10(split_index, row_vector);
        vector<vector<int>> a11(split_index, row_vector);

        vector<vector<int>> b00(split_index, row_vector);
        vector<vector<int>> b01(split_index, row_vector);
        vector<vector<int>> b10(split_index, row_vector);
```

```
    vector<vector<int>> b11(split_index, row_vector);


    for (auto i = 0; i < split_index; i++)
       for (auto j = 0; j < split_index; j++)
       {
          a00[i][j] = A[i][j];
          a01[i][j] = A[i][j + split_index];
          a10[i][j] = A[split_index + i][j];
          a11[i][j] = A[i + split_index][j + split_index];
          b00[i][j] = B[i][j];
          b01[i][j] = B[i][j + split_index];
          b10[i][j] = B[split_index + i][j];
          b11[i][j] = B[i + split_index][j + split_index];
       }


    vector<vector<int>> p1(strassen_multiplication(a00, add(b01, b11, split_index, -1)));
    vector<vector<int>> p2(strassen_multiplication(add(a00, a01, split_index), b11));
    vector<vector<int>> p3(strassen_multiplication(add(a10, a11, split_index), b00));
    vector<vector<int>> p4(strassen_multiplication(a11, add(b10, b00, split_index, -1)));
    vector<vector<int>> p5(strassen_multiplication(add(a00, a11, split_index), add(b00, b11,
split_index)));
    vector<vector<int>> p6(strassen_multiplication(add(a01, a11, split_index, -1), add(b10, b11,
split_index)));
    vector<vector<int>> p7(strassen_multiplication(add(a00, a10, split_index, -1), add(b00, b01,
split_index)));


    vector<vector<int>> result_00(add(add(add(p5, p4, split_index), p6, split_index), p2, split_index, -
1));
    vector<vector<int>> result_01(add(p1, p2, split_index));
    vector<vector<int>> result_10(add(p3, p4, split_index));
    vector<vector<int>> result_11(add(add(add(p5, p1, split_index), p3, split_index, -1), p7, split_index,
-1));


    for (auto i = 0; i < split_index; i++)
```

```
    {
        for (auto j = 0; j < split_index; j++)
        {
            result[i][j] = result_00[i][j];
            result[i][j + split_index] = result_01[i][j];
            result[split_index + i][j] = result_10[i][j];
            result[i + split_index][j + split_index] = result_11[i][j];
        }
    }

    a00.clear();
    a01.clear();
    a10.clear();
    a11.clear();
    b00.clear();
    b01.clear();
    b10.clear();
    b11.clear();
    p1.clear();
    p2.clear();
    p3.clear();
    p4.clear();
    p5.clear();
    p6.clear();
    p7.clear();
    result_00.clear();
    result_01.clear();
    result_10.clear();
    result_11.clear();
}

return result;
```

```
}
// Function to print each step of the matrix multiplication
void print_steps(vector<vector<int>> A, vector<vector<int>> B, vector<vector<int>> C)
{
    int n = A.size();
    int m = A[0].size();
    int p = B[0].size();

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            int sum = 0;
            cout << "Step for C[" << i << "][" << j << "]:" << endl;
            for (int k = 0; k < m; k++)
            {
                cout << "C[" << i << "][" << j << "] += A[" << i << "][" << k << "] * B[" << k << "][" << j <<
"] ";
                cout << "= " << A[i][k] << " * " << B[k][j] << " = " << A[i][k] * B[k][j] << endl;
                sum += A[i][k] * B[k][j];
            }
            cout << "C[" << i << "][" << j << "] = " << sum << endl
                 << endl;
        }
    }
}
int main()
{
    int row_1, col_1, row_2, col_2;

    cout << "Enter the number of rows and columns for Matrix A: ";
    cin >> row_1 >> col_1;
```

```
    cout << "Enter the number of rows and columns for Matrix B: ";
    cin >> row_2 >> col_2;

    if (col_1 != row_2)
    {
        cout << "The two matrices cannot be multiplied.";
        return 0;
    }

    vector<vector<int>> A(row_1, vector<int>(col_1));
    vector<vector<int>> B(row_2, vector<int>(col_2));

    cout << "Enter Matrix A:" << endl;
    for (int i = 0; i < row_1; i++)
        for (int j = 0; j < col_1; j++)
            cin >> A[i][j];

    cout << "Enter Matrix B:" << endl;
    for (int i = 0; i < row_2; i++)
        for (int j = 0; j < col_2; j++)
            cin >> B[i][j];

    cout << "Step-by-step multiplication:" << endl;
    print_steps(A, B, strassen_multiplication(A, B));

    cout << "Resultant Matrix:" << endl;
    print(strassen_multiplication(A, B));

    return 0;
}
```

Output-

```
Enter the number of rows and columns for Matrix A: 2 2
Enter the number of rows and columns for Matrix B: 2 2
Enter Matrix A:
4 5 2 7
Enter Matrix B:
4 1 6 8
Step-by-step multiplication:
Step for C[0][0]:
C[0][0] += A[0][0] * B[0][0] = 4 * 4 = 16
C[0][0] += A[0][1] * B[1][0] = 5 * 6 = 30
C[0][0] = 46

Step for C[0][1]:
C[0][1] += A[0][0] * B[0][1] = 4 * 1 = 4
C[0][1] += A[0][1] * B[1][1] = 5 * 8 = 40
C[0][1] = 44

Step for C[1][0]:
C[1][0] += A[1][0] * B[0][0] = 2 * 4 = 8
C[1][0] += A[1][1] * B[1][0] = 7 * 6 = 42
C[1][0] = 50

Step for C[1][1]:
C[1][1] += A[1][0] * B[0][1] = 2 * 1 = 2
C[1][1] += A[1][1] * B[1][1] = 7 * 8 = 56
C[1][1] = 58

Resultant Matrix:
46 44
50 58
```