

Program No: 01

Program statement: Matrix Multiplication Using Naive Approach

Program Code:

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

int main() {
    int m, n, p;
    cout << "Enter dimensions (m n p): ";
    cin >> m >> n >> p;

    int A[m][n], B[n][p], C[m][p];

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

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

    for (int i = 0; i < m; i++)
        for (int j = 0; j < p; j++)
            C[i][j] = 0;

    for (int i = 0; i < m; i++)
        for (int j = 0; j < p; j++)
            for (int k = 0; k < n; k++)
                C[i][j] += A[i][k] * B[k][j];

    cout << "Result:" << endl;
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < p; j++)
            cout << C[i][j] << " ";
        cout << endl;
    }

    return 0;
}
```

Program Output:

```
"F:\01. Computer-Algorithm-Sessional\01. Academic\LAB_06\problem_01.exe"
Enter dimensions (m n p): 3 3 3
Enter Matrix A:
1 2 3
4 5 6
7 8 9
Enter Matrix B:
1 3 5
7 9 1
3 5 7
Result:
24 36 28
57 87 67
90 138 106
```

Program No: 02

Program statement: Matrix Multiplication Using Divide and Conquer Approach

Program Code:

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

typedef vector<vector<int>> Matrix;

Matrix add(const Matrix &A, const Matrix &B) {
    int n = A.size();
    Matrix C(n, vector<int>(n));
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            C[i][j] = A[i][j] + B[i][j];
    return C;
}

Matrix subtract(const Matrix &A, const Matrix &B) {
    int n = A.size();
    Matrix C(n, vector<int>(n));
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            C[i][j] = A[i][j] - B[i][j];
    return C;
}
```

```

Matrix multiply(const Matrix &A, const Matrix &B) {
    int n = A.size();
    Matrix C(n, vector<int>(n, 0));
    if (n == 1) {
        C[0][0] = A[0][0] * B[0][0];
    } else {
        int k = n / 2;
        Matrix A11(k, vector<int>(k)), A12(k, vector<int>(k)),
            A21(k, vector<int>(k)), A22(k, vector<int>(k));
        Matrix B11(k, vector<int>(k)), B12(k, vector<int>(k)),
            B21(k, vector<int>(k)), B22(k, vector<int>(k));
        for (int i = 0; i < k; i++)
            for (int j = 0; j < k; j++) {
                A11[i][j] = A[i][j];
                A12[i][j] = A[i][j + k];
                A21[i][j] = A[i + k][j];
                A22[i][j] = A[i + k][j + k];
                B11[i][j] = B[i][j];
                B12[i][j] = B[i][j + k];
                B21[i][j] = B[i + k][j];
                B22[i][j] = B[i + k][j + k];
            }

        Matrix C11 = add(multiply(A11, B11), multiply(A12, B21));
        Matrix C12 = add(multiply(A11, B12), multiply(A12, B22));
        Matrix C21 = add(multiply(A21, B11), multiply(A22, B21));
        Matrix C22 = add(multiply(A21, B12), multiply(A22, B22));

        for (int i = 0; i < k; i++)
            for (int j = 0; j < k; j++) {
                C[i][j] = C11[i][j];
                C[i][j + k] = C12[i][j];
                C[i + k][j] = C21[i][j];
                C[i + k][j + k] = C22[i][j];
            }
    }
    return C;
}

```

```

int main() {
    int n;
    cout << "Enter matrix size (power of 2): ";
    cin >> n;

    Matrix A(n, vector<int>(n)), B(n, vector<int>(n));

    cout << "Enter Matrix A:" << endl;
}

```

```

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

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

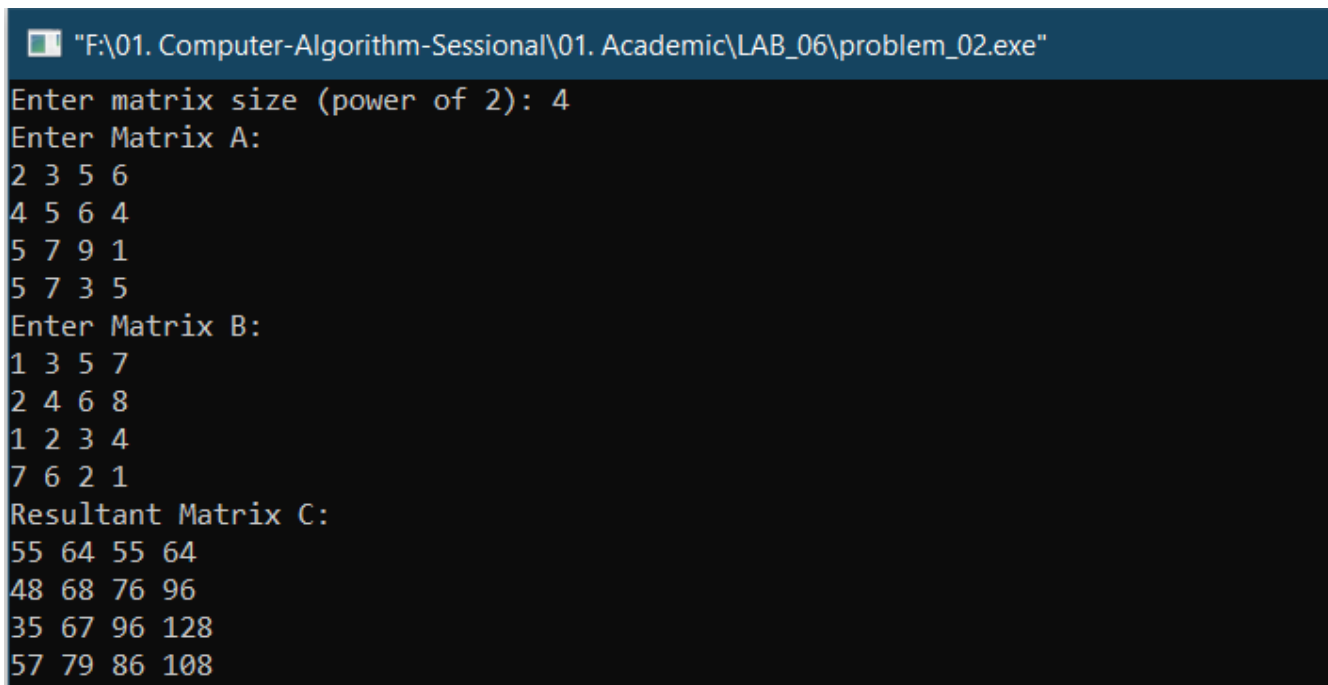
Matrix C = multiply(A, B);

cout << "Resultant Matrix C:" << endl;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++)
        cout << C[i][j] << " ";
    cout << endl;
}

return 0;
}

```

Program Output:



```

"F:\01. Computer-Algorithm-Sessional\01. Academic\LAB_06\problem_02.exe"
Enter matrix size (power of 2): 4
Enter Matrix A:
2 3 5 6
4 5 6 4
5 7 9 1
5 7 3 5
Enter Matrix B:
1 3 5 7
2 4 6 8
1 2 3 4
7 6 2 1
Resultant Matrix C:
55 64 55 64
48 68 76 96
35 67 96 128
57 79 86 108

```

Program No: 03

Program statement: Matrix Multiplication Using Strassen's Algorithm

Program Code:

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

typedef vector<vector<int>> Matrix;

Matrix add(const Matrix &A, const Matrix &B) {
    int n = A.size();
    Matrix C(n, vector<int>(n));
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            C[i][j] = A[i][j] + B[i][j];
    return C;
}

Matrix subtract(const Matrix &A, const Matrix &B) {
    int n = A.size();
    Matrix C(n, vector<int>(n));
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            C[i][j] = A[i][j] - B[i][j];
    return C;
}

Matrix strassen(const Matrix &A, const Matrix &B) {
    int n = A.size();
    Matrix C(n, vector<int>(n, 0));

    if (n == 1) {
        C[0][0] = A[0][0] * B[0][0];
        return C;
    }

    int k = n / 2;
    Matrix A11(k, vector<int>(k)), A12(k, vector<int>(k)),
        A21(k, vector<int>(k)), A22(k, vector<int>(k));
    Matrix B11(k, vector<int>(k)), B12(k, vector<int>(k)),
        B21(k, vector<int>(k)), B22(k, vector<int>(k));

    for (int i = 0; i < k; i++) {
        for (int j = 0; j < k; j++) {
            A11[i][j] = A[i][j];
            A12[i][j] = A[i][j + k];
```

```

        A21[i][j] = A[i + k][j];
        A22[i][j] = A[i + k][j + k];
        B11[i][j] = B[i][j];
        B12[i][j] = B[i][j + k];
        B21[i][j] = B[i + k][j];
        B22[i][j] = B[i + k][j + k];
    }
}

```

```

Matrix M1 = strassen(add(A11, A22), add(B11, B22));
Matrix M2 = strassen(add(A21, A22), B11);
Matrix M3 = strassen(A11, subtract(B12, B22));
Matrix M4 = strassen(A22, subtract(B21, B11));
Matrix M5 = strassen(add(A11, A12), B22);
Matrix M6 = strassen(subtract(A21, A11), add(B11, B12));
Matrix M7 = strassen(subtract(A12, A22), add(B21, B22));

```

```

Matrix C11 = add(subtract(add(M1, M4), M5), M7);
Matrix C12 = add(M3, M5);
Matrix C21 = add(M2, M4);
Matrix C22 = add(subtract(add(M1, M3), M2), M6);

```

```

for (int i = 0; i < k; i++) {
    for (int j = 0; j < k; j++) {
        C[i][j] = C11[i][j];
        C[i][j + k] = C12[i][j];
        C[i + k][j] = C21[i][j];
        C[i + k][j + k] = C22[i][j];
    }
}

```

```

return C;

```

```

}

```

```

int main() {
    int n;
    cout << "Enter matrix size (power of 2): ";
    cin >> n;

    Matrix A(n, vector<int>(n)), B(n, vector<int>(n));

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

    cout << "Enter Matrix B:" << endl;
    for (int i = 0; i < n; i++)

```

```

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

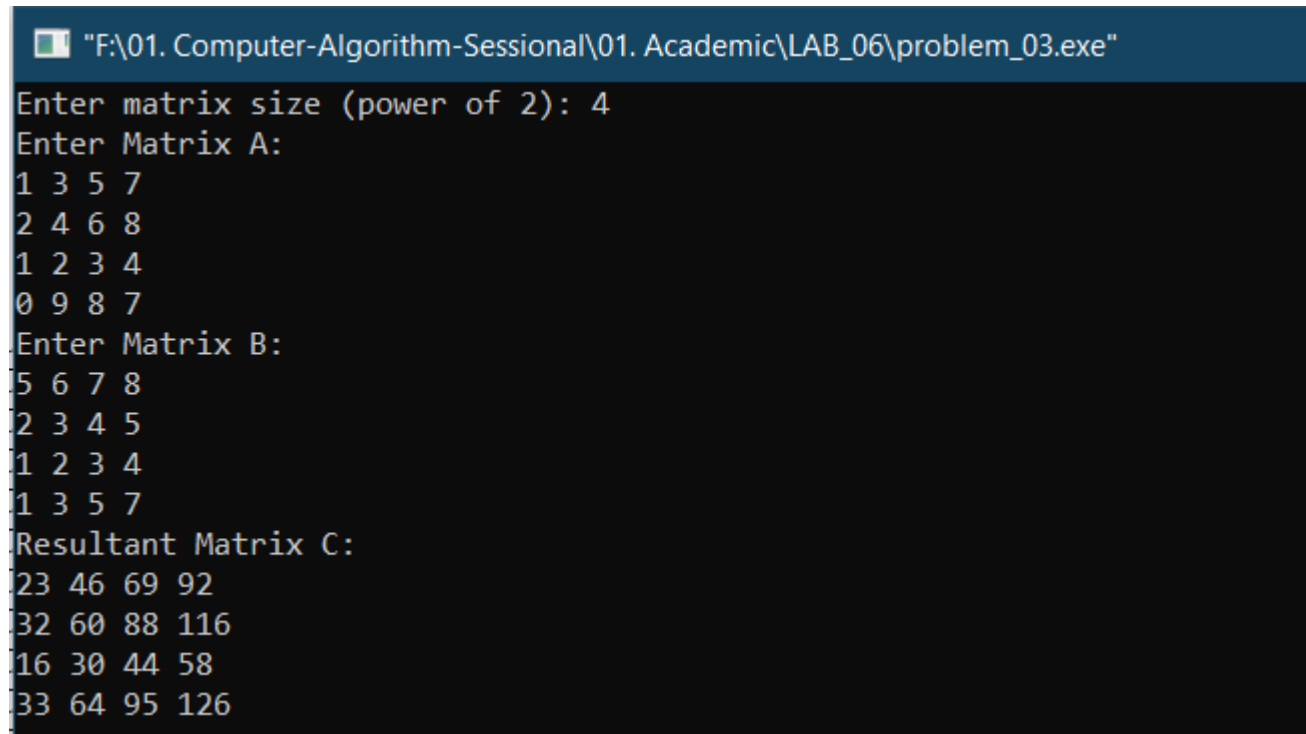
Matrix C = strassen(A, B);

cout << "Resultant Matrix C:" << endl;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++)
        cout << C[i][j] << " ";
    cout << endl;
}

return 0;
}

```

Program Output:



```

F:\01. Computer-Algorithm-Sessional\01. Academic\LAB_06\problem_03.exe
Enter matrix size (power of 2): 4
Enter Matrix A:
1 3 5 7
2 4 6 8
1 2 3 4
0 9 8 7
Enter Matrix B:
5 6 7 8
2 3 4 5
1 2 3 4
1 3 5 7
Resultant Matrix C:
23 46 69 92
32 60 88 116
16 30 44 58
33 64 95 126

```