Write c++ code for Strassens Matrix

#include <bits/stdc++.h>

using namespace std;

typedef long long lld;

/\* Strassen's Algorithm for matrix multiplication

Complexity: O(n^2.808) \*/

inline lld\*\* MatrixMultiply(lld\*\* a, lld\*\* b, int n,

                                    int l, int m)

{

    lld\*\* c = new lld\*[n];

    for (int i = 0; i < n; i++)

        c[i] = new lld[m];

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < m; j++) {

            c[i][j] = 0;

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

                c[i][j] += a[i][k] \* b[k][j];

            }

        }

    }

    return c;

}

inline lld\*\* Strassen(lld\*\* a, lld\*\* b, int n,

                                int l, int m)

{

    if (n == 1 || l == 1 || m == 1)

        return MatrixMultiply(a, b, n, l, m);

    lld\*\* c = new lld\*[n];

    for (int i = 0; i < n; i++)

        c[i] = new lld[m];

    int adjN = (n >> 1) + (n & 1);

    int adjL = (l >> 1) + (l & 1);

    int adjM = (m >> 1) + (m & 1);

    lld\*\*\*\* As = new lld\*\*\*[2];

    for (int x = 0; x < 2; x++) {

        As[x] = new lld\*\*[2];

        for (int y = 0; y < 2; y++) {

            As[x][y] = new lld\*[adjN];

            for (int i = 0; i < adjN; i++) {

                As[x][y][i] = new lld[adjL];

                for (int j = 0; j < adjL; j++) {

                    int I = i + (x & 1) \* adjN;

                    int J = j + (y & 1) \* adjL;

                    As[x][y][i][j] = (I < n && J < l) ? a[I][J] : 0;

                }

            }

        }

    }

    lld\*\*\*\* Bs = new lld\*\*\*[2];

    for (int x = 0; x < 2; x++) {

        Bs[x] = new lld\*\*[2];

        for (int y = 0; y < 2; y++) {

            Bs[x][y] = new lld\*[adjN];

            for (int i = 0; i < adjL; i++) {

                Bs[x][y][i] = new lld[adjM];

                for (int j = 0; j < adjM; j++) {

                    int I = i + (x & 1) \* adjL;

                    int J = j + (y & 1) \* adjM;

                    Bs[x][y][i][j] = (I < l && J < m) ? b[I][J] : 0;

                }

            }

        }

    }

    lld\*\*\* s = new lld\*\*[10];

    for (int i = 0; i < 10; i++) {

        switch (i) {

        case 0:

            s[i] = new lld\*[adjL];

            for (int j = 0; j < adjL; j++) {

                s[i][j] = new lld[adjM];

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

                    s[i][j][k] = Bs[0][1][j][k] - Bs[1][1][j][k];

                }

            }

            break;

        case 1:

            s[i] = new lld\*[adjN];

            for (int j = 0; j < adjN; j++) {

                s[i][j] = new lld[adjL];

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

                    s[i][j][k] = As[0][0][j][k] + As[0][1][j][k];

                }

            }

            break;

        case 2:

            s[i] = new lld\*[adjN];

            for (int j = 0; j < adjN; j++) {

                s[i][j] = new lld[adjL];

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

                    s[i][j][k] = As[1][0][j][k] + As[1][1][j][k];

                }

            }

            break;

        case 3:

            s[i] = new lld\*[adjL];

            for (int j = 0; j < adjL; j++) {

                s[i][j] = new lld[adjM];

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

                    s[i][j][k] = Bs[1][0][j][k] - Bs[0][0][j][k];

                }

            }

            break;

        case 4:

            s[i] = new lld\*[adjN];

            for (int j = 0; j < adjN; j++) {

                s[i][j] = new lld[adjL];

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

                    s[i][j][k] = As[0][0][j][k] + As[1][1][j][k];

                }

            }

            break;

        case 5:

            s[i] = new lld\*[adjL];

            for (int j = 0; j < adjL; j++) {

                s[i][j] = new lld[adjM];

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

                    s[i][j][k] = Bs[0][0][j][k] + Bs[1][1][j][k];

                }

            }

            break;

        case 6:

            s[i] = new lld\*[adjN];

            for (int j = 0; j < adjN; j++) {

                s[i][j] = new lld[adjL];

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

                    s[i][j][k] = As[0][1][j][k] - As[1][1][j][k];

                }

            }

            break;

        case 7:

            s[i] = new lld\*[adjL];

            for (int j = 0; j < adjL; j++) {

                s[i][j] = new lld[adjM];

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

                    s[i][j][k] = Bs[1][0][j][k] + Bs[1][1][j][k];

                }

            }

            break;

        case 8:

            s[i] = new lld\*[adjN];

            for (int j = 0; j < adjN; j++) {

                s[i][j] = new lld[adjL];

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

                    s[i][j][k] = As[0][0][j][k] - As[1][0][j][k];

                }

            }

            break;

        case 9:

            s[i] = new lld\*[adjL];

            for (int j = 0; j < adjL; j++) {

                s[i][j] = new lld[adjM];

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

                    s[i][j][k] = Bs[0][0][j][k] + Bs[0][1][j][k];

                }

            }

            break;

        }

    }

    lld\*\*\* p = new lld\*\*[7];

    p[0] = Strassen(As[0][0], s[0], adjN, adjL, adjM);

    p[1] = Strassen(s[1], Bs[1][1], adjN, adjL, adjM);

    p[2] = Strassen(s[2], Bs[0][0], adjN, adjL, adjM);

    p[3] = Strassen(As[1][1], s[3], adjN, adjL, adjM);

    p[4] = Strassen(s[4], s[5], adjN, adjL, adjM);

    p[5] = Strassen(s[6], s[7], adjN, adjL, adjM);

    p[6] = Strassen(s[8], s[9], adjN, adjL, adjM);

    for (int i = 0; i < adjN; i++) {

        for (int j = 0; j < adjM; j++) {

            c[i][j] = p[4][i][j] + p[3][i][j] - p[1][i][j] + p[5][i][j];

            if (j + adjM < m)

                c[i][j + adjM] = p[0][i][j] + p[1][i][j];

            if (i + adjN < n)

                c[i + adjN][j] = p[2][i][j] + p[3][i][j];

            if (i + adjN < n && j + adjM < m)

                c[i + adjN][j + adjM] = p[4][i][j] + p[0][i][j] - p[2][i][j] - p[6][i][j];

        }

    }

    for (int x = 0; x < 2; x++) {

        for (int y = 0; y < 2; y++) {

            for (int i = 0; i < adjN; i++) {

                delete[] As[x][y][i];

            }

            delete[] As[x][y];

        }

        delete[] As[x];

    }

    delete[] As;

    for (int x = 0; x < 2; x++) {

        for (int y = 0; y < 2; y++) {

            for (int i = 0; i < adjL; i++) {

                delete[] Bs[x][y][i];

            }

            delete[] Bs[x][y];

        }

        delete[] Bs[x];

    }

    delete[] Bs;

    for (int i = 0; i < 10; i++) {

        switch (i) {

        case 0:

        case 3:

        case 5:

        case 7:

        case 9:

            for (int j = 0; j < adjL; j++) {

                delete[] s[i][j];

            }

            break;

        case 1:

        case 2:

        case 4:

        case 6:

        case 8:

            for (int j = 0; j < adjN; j++) {

                delete[] s[i][j];

            }

            break;

        }

        delete[] s[i];

    }

    delete[] s;

    for (int i = 0; i < 7; i++) {

        for (int j = 0; j < (n >> 1); j++) {

            delete[] p[i][j];

        }

        delete[] p[i];

    }

    delete[] p;

    return c;

}

int main()

{

    lld\*\* matA;

    matA = new lld\*[2];

    for (int i = 0; i < 2; i++)

        matA[i] = new lld[3];

    matA[0][0] = 1;

    matA[0][1] = 2;

    matA[0][2] = 3;

    matA[1][0] = 4;

    matA[1][1] = 5;

    matA[1][2] = 6;

    lld\*\* matB;

    matB = new lld\*[3];

    for (int i = 0; i < 3; i++)

        matB[i] = new lld[2];

    matB[0][0] = 7;

    matB[0][1] = 8;

    matB[1][0] = 9;

    matB[1][1] = 10;

    matB[2][0] = 11;

    matB[2][1] = 12;

    lld\*\* matC = Strassen(matA, matB, 2, 3, 2);

    for (int i = 0; i < 2; i++) {

        for (int j = 0; j < 2; j++) {

            printf("%lld ", matC[i][j]);

        }

        printf("\n");

    }

    return 0;

}