```
//MCM
#include <iostream>
#include <vector>
#include <climits>
using namespace std;
struct Matrix {
    int rows, cols;
    Matrix(int r = 0, int c = 0) : rows(r), cols(c) {}
};

vector<Matrix> matrices;
vector<vector<int> > m;
vector<vector<int> > s;

void printOptimalOrder(int i, int j) {
    if (i == j) {
        cout << "A" << i;
    } else {
        cout << "(";
        printOptimalOrder(i, s[i][j]);
        cout << " x ";
        printOptimalOrder(s[i][j] + 1, j);
        cout << ")";
    }
}

void matrixChainOrder(int n) {
    m.assign(n, vector<int>(n, 0));
    s.assign(n, vector<int>(n, 0));

    for (int l = 2; l < n; l++) {
        for (int i = 1; i < n - l + 1; i++) {
            int j = i + l - 1;
            m[i][j] = INT_MAX;

            for (int k = i; k < j; k++) {
                int cost = m[i][k] + m[k + 1][j] + matrices[i - 1].rows *
matrices[k].cols * matrices[j].cols;
                if (cost < m[i][j]) {
                    m[i][j] = cost;
                    s[i][j] = k;
                }
            }
        }
    }
}

int main() {
    int n;
    cin >> n;

    matrices.resize(n + 1);
    for (int i = 0; i < n; i++) {
        int rows, cols;
        cin >> rows >> cols;
        matrices[i] = Matrix(rows, cols);
    }

    matrixChainOrder(n);

    cout << "Minimum cost for matrix multiplication: " << m[1][n - 1] << endl;
    cout << "Optimal order for matrix multiplication: ";
    printOptimalOrder(1, n - 1);
    cout << endl;

    return 0;
}
```

```
//Topological sort
#include <bits/stdc++.h>
using namespace std;
```

```
void topo_sort(int vertices,
    int edges) {
        vector<char> ans;
        queue<char> q;
        map<char, vector<char>>graph;
        map<char, int> inDegree;
        vector<pair<char, char>>
        edgeList = {{'A', 'B'},{'A', 'C'},{'B', 'D'},{'B', 'E'},{'C', 'E'},{'D',
'F'},{'E', 'F'}};
        for (int i = 0; i < edges; i++){
            char a = edgeList[i].first;
            char b =edgeList[i].second;
            graph[a].push_back(b);
            inDegree[b]++;
        }
        for (char c = 'A'; c <= 'F';c++) {
            if (inDegree[c] == 0) {
                q.push(c);
            }
        }
    while (!q.empty()) {
        char v = q.front();
        q.pop();
        ans.push_back(v);
        for (int i = 0; i <graph[v].size(); i++) {
            char u = graph[v][i];
            inDegree[u]--;
            if (inDegree[u] == 0) {
                q.push(u);
            }
        }
    }
    for (int i = 0; i < ans.size();i++) {
        cout << ans[i];
        if (i < ans.size() - 1) {
                cout << " ";
            }
        }
    }
int main() {
    int vertices = 6;
    int edges = 7;
    topo_sort(vertices, edges);
    return 0;
}
```