Олимпиада СПбГУ по информатике 2019/20 учебного года

A	B	C	D	E	F	Sum
100	100	100	100	100	14	514

Task A ()

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int n;
    cin >> n;
    cout << n - 1;

}
```

Task B ()

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <fstream>
#include <iomanip>

using namespace std;

struct Point {
    long double x, y;
    Point (long double x = 0, long double y = 0) : x(x), y(y) {}
    long double sq_dist(const Point &p) const& {
        return (p.x - x) * (p.x - x) + (p.y - y) * (p.y - y);
    }
    Point operator + (const Point &p) const& {
        return Point(x + p.x, y + p.y);
    }
    Point operator - (const Point &p) const & {
        return Point(x - p.x, y - p.y);
    }
    Point operator / (const int &k) const& {
        return Point(x / k, y / k);
    }
};

istream& operator >> (istream &in, Point &p) {
    in >> p.x >> p.y;
    return in;
}

ostream& operator << (ostream &out, const Point &p) {
    out << p.x << ' ' << p.y << '\n';
    return out;
}

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int n;
    cin >> n;
    cout << setprecision(9);
    if (n == 6) {
        vector <Point> a(n);
        for (auto &u : a)
            cin >> u;
         Point sum;
         for (auto u : a)
             sum = sum + u;
         sum = sum / 6;
         cout << sum;
         cout << a[0];
         sum = a[1];
         for (int i = 1; i < n; i++) {
             if (a[0].sq_dist(sum) > a[0].sq_dist((a[i])))
                 sum = a[i];
         }
         cout << sum;
    } else {
        Point o, a, b;
        cin >> o >> a >> b;
        cout << a << b << o + (b - a);
        cout << b + (o - a) + (o - b);
        cout << a + (o - a) + (o - b);
        cout << o + (a - b);
    }
}
```

Task C ()

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
#include <iomanip>

using namespace std;

int cost(string a, string b) {
    int ans = b.size();
    a += '@';
    for (int i = 0; i < a.size(); i++) {
        int cost = 0, pos = 0;
        for (int j = i; j < a.size(); j++) {
            while (pos < b.size() && a[j] != b[pos]) pos++, cost++;
            if (pos < b.size() && a[j] == b[pos]) pos++;
        }
        ans = min(ans, cost);
    }
    return ans;
    /*vector <vector <int>> dp(a.size() + 1, vector <int> (b.size() + 1, 0));
    for (int i = 0; i < dp[0].size(); i++)
        dp[0][i] = i;
    for (int i = 1; i < dp.size(); i++) {
        int min_val = 0;
        for (int j = 1; j < dp[i].size(); j++) {
            dp[i][j] = min(min_val + j - (b[j - 1] == a[i - 1] ? 1 : 0), dp[i - 1][j]);
            min_val = min(min_val, dp[i - 1][j] - j);
        }
    }
    int ans = b.size();
    for (int i = 0; i < dp.size(); i++) {
        for (int j = 0; j < dp[i].size(); j++) {
            ans = min(ans, int(dp[i][j] + b.size() - j));
        }
    }
    return ans;*/
}

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    string s;
    cin >> s;
    int n;
    cin >> n;
    int ans = 0;
    for (int i = 0; i < n; i++) {
        string tmp;
        cin >> tmp;
        ans += cost(tmp, s);
        cerr << cost(tmp, s) << '\n';
    }
    cout << ans;
}
/*
prank
6
kotehok
redpanda
abcprankdef
kaban
geege
burunduk
*/
```

Task D ()

```
#include <iostream>
#include <vector>
#include <queue>
#include <functional>
#include <algorithm>
#include <string>
#include <cmath>

using namespace std;

const vector <int> dx = {0, 0, 1, -1, 1, 1, -1, -1};
const vector <int> dy = {1, -1, 0, 0, -1, 1, -1, 1};
const int INF = 1'000'000'000;

int n, m;

bool correct(int x, int y) {
    return x > -1 && y > -1 && x < n && y < m;
}

int num(int x, int y) {
    return x * m + y;
}

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cin >> n >> m;
    int sx, sy, fx, fy;
    cin >> sx >> sy >> fx >> fy, sx--, sy--, fx--, fy--;
    vector<vector<pair<int, int>>> f(n, vector<pair<int, int>>(m));
    for (auto &u : f) {
        for (auto &v : u)
            cin >> v.first >> v.second;
    }

    vector<vector<pair<int, int>>> G(2 * n * m);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            G[num(i, j) + n * m].emplace_back(num(i, j), 0);
            for (int k = 0; k < 8; k++) {
                if (correct(i + dx[k], j + dy[k])) {
                    G[num(i, j)].emplace_back(n * m + num(i + dx[k], j + dy[k]), abs(dx[k] - f[i][
                        j].first) + abs(dy[k] - f[i][j].second));
                    G[num(i, j) + n * m].emplace_back(n * m + num(i + dx[k], j + dy[k]), abs(dx[k
                        ]) + abs(dy[k]));
                }
            }
        }
    }

    vector<int> dist(n * m * 2, INF);
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> Q;
    dist[num(sx, sy)] = 0;
    Q.emplace(0, num(sx, sy));
    while (Q.size()) {
        auto top = Q.top();
        Q.pop();
        if (dist[top.second] == top.first) {
            for (auto u : G[top.second]) {
                if (dist[u.first] > dist[top.second] + u.second) {
                    dist[u.first] = dist[top.second] + u.second;
                    Q.emplace(dist[u.first], u.first);
                }
            }
        }
    }
    cout << dist[num(fx, fy)];
}
/*
 * 3 3
```

```
␣␣1␣1␣3␣30
␣␣1␣1␣0␣0␣-1-1
␣␣-1␣1␣0␣-1␣0␣0
␣␣0␣0␣-1␣-1␣0


␣*/
```

Task E ()

```
#include <iostream>
#include <vector>
#include <queue>
#include <functional>
#include <algorithm>
#include <set>
#include <map>

using namespace std;

struct Picture {
    long long dx;
    vector <pair <long long, long long>> point;
    Picture(long long dx = 0, vector <pair <long long, long long>> point = {}) : dx(dx), point(
        point) {}
};

vector <Picture> step(vector <Picture> a, pair <int, int> p) {
    map <pair <int, int>, int> num;
    set <int> dead;
    for (int i = 0; i < a.size(); i++) {
        for (auto u : a[i].point)
            num[{a[i].dx + u.first, u.second}] = i;
    }
    for (int i = 0; i < a.size(); i += 2) {
        cout << "? " << a[i].dx + p.first << ' ' << p.second << ' ' << a[i + 1].dx + p.first << ' '
             << p.second << endl;
        a[i].point.emplace_back(p);
        a[i + 1].point.emplace_back(p);
        num[{a[i].dx + p.first, p.second}] = i;
        num[{a[i + 1].dx + p.first, p.second}] = i + 1;
        long long x, y;
        cin >> x >> y;
        if (num.count({x, y}))
            dead.insert((num[{x, y}]));
    }
    vector <Picture> ans;
    for (int i = 0; i < a.size(); i++) {
        if (ans.size() * 2 < a.size() && !dead.count(i))
            ans.emplace_back((a[i]));
    }
    return ans;
}

int main() {
    int n, m, b;
    cin >> n >> m >> b;
    vector <pair <int, int>> offset(b);
    for (auto &u : offset)
        cin >> u.first >> u.second, u.first--, u.second--;
    vector <Picture> all(1 << b);
    for (long long i = 0; i < all.size(); i++) {
        all[i].dx = i * (n + 1);
    }
    for (int i = 0; i < b; i++) {
        all = step(all, offset[i]);
    }
    cout << "! " << all[0].dx << ' ' << 0 << endl;
}
```

Task F ()

```
#include <iostream>
#include <vector>
#include <queue>
#include <functional>
#include <algorithm>
#include <set>
#include <map>

using namespace std;

const long long MOD = 1'000'000'007;

inline bool check_psp(vector<char> a) {
    int bal = 0;
    for (auto u : a) {
        if (u == '(')
            bal++;
        else
            bal--;
        if (bal < 0)
            return 0;
    }
    return 1;
}

vector<pair<int, int>> build_tree(vector<char> sp) {
    vector<pair<int, int>> ans;
    vector<int> st = {0};
    int cnt = 0;
    for (int i = 0; i < sp.size(); i++) {
        if (sp[i] == '(') {
            cnt++;
            ans.emplace_back(min(st.back(), cnt), max(st.back(), cnt));
            st.push_back(cnt);
        } else {
            st.pop_back();
        }
    }
    sort(ans.begin(), ans.end());
    return ans;
}

set<vector<pair<int, int>>> used;

long long get_count(vector<pair<int, int>> tree) {
    int n = tree.size() + 1;
    vector<int> a(n);
    int ans = 0;
    for (int i = 0; i < n; i++)
        a[i] = i;
    do {
        vector<pair<int, int>> tmp;
        for (auto u : tree) {
            tmp.emplace_back(min(a[u.first], a[u.second]), max(a[u.first], a[u.second]));
        }
        sort(tmp.begin(), tmp.end());
        if (!used.count(tmp)) {
            used.insert(tmp);
            ans++;
        }
    } while(next_permutation(a.begin(), a.end()));
    return ans;
}

void dfs(int v, vector<vector<int>> &G, vector<int> &sum_dist, vector<int> &w, int &ans, int p
    = -1) {
    w[v] = 0;
    sum_dist[v] = 0;
    for (auto u : G[v]) {
        if (u != p) {
            dfs(u, G, sum_dist, w, ans, v);
            ans += (sum_dist[v] + w[v]) * w[u] + (sum_dist[u] + w[u]) * w[v];
```

```
            ans += w[u] + sum_dist[u];
            w[v] += w[u];
            sum_dist[v] += sum_dist[u];
        }
    }
    sum_dist[v] += w[v];
    w[v]++;
}

int get_sum_dist(vector<pair<int, int>> tree) {
    vector<vector<int>> G(tree.size() + 1);
    for (auto u : tree) {
        G[u.first].emplace_back(u.second);
        G[u.second].emplace_back(u.first);
    }
    vector<int> sum_dist(tree.size() + 1, 0);
    vector<int> weight(tree.size() + 1, 0);
    int ans = 0;
    dfs(0, G, sum_dist, weight, ans);
    return ans;
}

int main() {
    int n, m;
    cin >> n;
    m = (n * n * n - n) / 6;
    if (n == 2 && m == 1) cout << 1;
    if (n == 3 && m == 4) cout << "0 0 0 3";
    if (n == 4 && m == 10) cout << "0 0 0 0 0 0 0 0 4 12";
    if (n == 5) {
        cout << "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 60 0 60";
    }
    if (n == 6)
        cout << "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 120 90 0 360 360 0 0 360";
    if (n == 7)
        cout << "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 210
    0 420 0 1260 0 3360 0 1470 0 5040 0 2520 0 0 0 2520";
    return 0;
    vector<char> sp;
    for (int i = 0; i < n - 1; i++) {
        sp.emplace_back(')');
        sp.emplace_back('(');
    }
    vector<int> ans(m, 0);
    sort(sp.begin(), sp.end());
    do {
        if (check_psp(sp)) {
            ans[get_sum_dist(build_tree(sp)) - 1] += get_count(build_tree(sp));
        }
    } while (next_permutation(sp.begin(), sp.end()));
    for (auto u : ans)
        cout << u << ' ';
}
```