Олимпиада СПбГУ по информатике 2019/20 учебного года

A	B	C	D	E	F	Sum
100	100	100	100	100	23	523

Task A ()

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include <functional>
#include <numeric>
#include <vector>
#include <queue>
#include <set>
#include <map>
#include <bitset>
#include <random>
#include <cassert>

using namespace std;

signed main() {
        ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
        int n;
        cin >> n;
        cout << n - 1 << '\n';
        //system("pause");
        return 0;
}
```

Task B ()

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include <functional>
#include <numeric>
#include <vector>
#include <queue>
#include <set>
#include <map>
#include <bitset>
#include <random>
#include <cassert>

using namespace std;

#define ld long double

struct Point {
        ld x, y;

        Point() {}
        Point(ld _x, ld _y) : x(_x), y(_y) {}
};
istream& operator >> (istream &in, Point &A) { return in >> A.x >> A.y; }
ostream& operator<<(ostream &out, Point A) { return out << A.x << '␣' << A.y; }

pair<ld, ld> make_pair(Point A) { return{ A.x, A.y }; }

struct Vector {
        ld x, y;

        Vector() {}
        Vector(ld _x, ld _y) : x(_x), y(_y) {}
        Vector(Point A) : x(A.x), y(A.y) {}
        Vector(Point A, Point B) : x(B.x - A.x), y(B.y - A.y) {}
};
Point operator+(Point P, Vector a) { return{ P.x + a.x, P.y + a.y }; }
ld cross_product(Vector a, Vector b) { return a.x * b.y - a.y * b.x; }

const int N = 6;

Point p[N];

signed main() {
        ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
        cout.precision(47); cout << fixed;
        int n;
        cin >> n;
        for (int i = 0; i < n; ++i) {
                cin >> p[i];
        }
        if (n == 6) {
                sort(p, p + n, [](Point A, Point B) { return make_pair(A) < make_pair(B); });
                sort(p + 1, p + n, [&](Point A, Point B) { return cross_product(Vector(p[0], A),
                    Vector(p[0], B)) < 0; });
                cout << p[0] << '\n' << p[2] << '\n' << p[4] << '\n';
        }
        else {
                Point O((p[0].x + p[1].x + p[2].x) / 3, (p[0].y + p[1].y + p[2].y) / 3);
                cout << p[0] << '\n' << O + Vector(p[2], O) << '\n' << p[1] << '\n' << O + Vector(
                    p[0], O) << '\n' << p[2] << '\n' << O + Vector(p[1], O) << '\n';
        }
        //system("pause");
        return 0;
}
```

Task C ()

```cpp
#include <iostream>
#include <cmath>
#include <algorithm>
#include <functional>
#include <numeric>
#include <vector>
#include <queue>
#include <set>
#include <map>
#include <bitset>
#include <random>
#include <cassert>

using namespace std;

const int K = 501;
const int N = 1e4 + 1;

int dp[K][N];

int f(string &s, string &t) {
        for (int i = 1; i <= s.size(); ++i) {
                dp[i][0] = i;
                for (int j = 1; j <= t.size(); ++j) {
                        if (s[i - 1] == t[j - 1]) {
                                dp[i][j] = dp[i - 1][j - 1];
                        }
                        else {
                                dp[i][j] = dp[i - 1][j] + 1;
                        }
                }
        }
        int res = 1e9;
        for (int j = 0; j <= t.size(); ++j) {
                res = min(res, dp[s.size()][j]);
        }
        return res;
}

signed main() {
        ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
        int n;
        string s, t;
        cin >> s >> n;
        int ans = 0;
        for (int i = 0; i < n; ++i) {
                cin >> t;
                ans += f(s, t);
        }
        cout << ans << '\n';
        //system("pause");
        return 0;
}
```

Task D ()

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include <functional>
#include <numeric>
#include <vector>
#include <queue>
#include <set>
#include <map>
#include <bitset>
#include <random>
#include <cassert>

using namespace std;

const int N = 1007;
const int V = N * N * 3;

int n, m, s, t;
int r[N][N][2];
struct Edge { int v, u, w; };
vector<Edge> g[V];

int decode(int i, int j, int state) { return i * N + j << 1 ^ state; }

void add_edge(int v, int u, int w) {
        g[v].push_back({ v, u, w });
}

const int INF = 1e9;

bool used[V];
int dist[V];

void dijkstra(int s) {
        fill(dist, dist + V, INF);
        set<pair<int, int>> f;
        f.insert({ 0, s });
        dist[s] = 0;
        while (!f.empty()) {
                int v = f.begin()->second;
                f.erase(f.begin());
                used[v] = true;
                for (Edge e : g[v]) {
                        if (!used[e.u] && dist[e.u] > dist[e.v] + e.w) {
                                if (dist[e.u] != INF) {
                                        f.erase({ dist[e.u], e.u });
                                }
                                dist[e.u] = dist[e.v] + e.w;
                                f.insert({ dist[e.u], e.u });
                        }
                }
        }
}

signed main() {
        ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
        cin >> n >> m;
        int ar, ac, br, bc;
        cin >> ar >> ac >> br >> bc;
        s = decode(ar, ac, 0), t = decode(br, bc, 0);
        for (int i = 0; i <= n + 1; ++i) {
                for (int j = 0; j <= m + 1; ++j) {
                        /*if (decode(i, j, 1) == 11) {
                                system("pause");
                        }*/
                        if (i > 0 && i <= n && j > 0 && j <= m) {
                                cin >> r[i][j][0] >> r[i][j][1];
                                int ni = i + r[i][j][0], nj = j + r[i][j][1];
                                add_edge(decode(i, j, 1), decode(i, j, 0), 0);
                                //if (ni > 0 && ni <= n && nj > 0 && nj <= m) {
                                        add_edge(decode(i, j, 0), decode(ni, nj, 1), 0);
                                //}
                        }
                        int dx[4] = { -1, 0, +1, 0 };
                        int dy[4] = { 0, +1, 0, -1 };
                        for (int k = 0; k < 4; ++k) {
                                int ni = i + dx[k];
                                int nj = j + dy[k];
                                //if (ni > 0 && ni <= n && nj > 0 && nj <= m) {
                                        add_edge(decode(i, j, 1), decode(ni, nj, 1), 1);
                                //}
                        }
                }
        }
        dijkstra(s);
        cout << dist[t] << '\n';
        //system("pause");
        return 0;
}
```

Task E ()

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include <functional>
#include <numeric>
#include <vector>
#include <queue>
#include <set>
#include <map>
#include <bitset>
#include <random>
#include <cassert>

using namespace std;

#define int long long

const int B = 13;

int n, m, cnt;
struct Point { int x, y; } p[B];
set<pair<int, int>> f;
set<int> broken;
int kek[B];

bool ok(pair<int, int> a) {
        return broken.find(a.second) == broken.end() && kek[a.first] < (1 << cnt - a.first);
}

auto get() {
        auto res = *f.begin();
        f.erase(f.begin());
        while (!ok(res)) {
                res = *f.begin();
                f.erase(f.begin());
        }
        return res;
}

void check(pair<int, int> a) {
        if (a.first == cnt && ok(a)) {
                cout << "!␣" << a.second << '␣' << 0 << endl;
                //system("pause");
                exit(0);
        }
}

void ask(pair<int, int> a, pair<int, int> b) {
        cout << "?␣" << a.second + p[a.first].x << '␣' << p[a.first].y << '␣' << b.second + p[b.
            first].x << '␣' << p[b.first].y << endl;
        ++kek[a.first], ++kek[b.first];
        int x, y;
        cin >> x >> y;
        broken.insert(x / n * n);
        ++a.first, ++b.first;
        check(a), check(b);
        f.insert(a);
        f.insert(b);
}

signed main() {
        ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
        cin >> n >> m >> cnt;
        for (int i = 0; i < cnt; ++i) {
                cin >> p[i].x >> p[i].y;
                --p[i].x, --p[i].y;
        }
        for (int i = 0; i < (1 << cnt); ++i) {
                f.insert({ 0, n * i });
        }
        while (true) {
                auto a = get();
                auto b = get();
                ask(a, b);
        }
        //system("pause");
        return 0;
}
```

Task F ()

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include <functional>
#include <numeric>
#include <vector>
#include <queue>
#include <set>
#include <map>
#include <bitset>
#include <random>
#include <cassert>

using namespace std;

#define int long long

const int N = 26;

int n, m;
vector<int> g[N];
int ans[(N * N * N - N) / 6 + 1];

int h[N];

void dfs(int v, int last = -1) {
        if (last != -1) {
                h[v] = h[last] + 1;
        }
        else {
                h[v] = 0;
        }
        for (int u : g[v]) {
                if (u != last) {
                        dfs(u, v);
                }
        }
}

int f() {
        int res = 0;
        for (int v = 0; v < n; ++v) {
                dfs(v);
                for (int u = 0; u < n; ++u) {
                        res += h[u];
                }
        }
        return res / 2;
}

vector<int> stack;

void rec(int mask) {
        if (mask == (1 << n) - 1) {
                ++ans[f()];
        }
        else {
                bool flag = false;
                for (int v = 0; v < n; ++v) {
                        if (~mask >> v & 1) {
                                for (int i = stack.size() - 1; i >= 0; --i) {
                                        g[v].push_back(stack[i]);
                                        g[stack[i]].push_back(v);
                                        stack.push_back(v);
                                        rec(mask ^ 1 << v);
                                        stack.pop_back();
                                        g[stack[i]].pop_back();
                                        g[v].pop_back();
                                        if (stack[i] > v) {
                                                break;
                                        }
                                }
                        }
                }
        }
}

signed main() {
        ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
        cin >> n >> m;
        stack.push_back(0);
        rec(1);
        for (int i = 1; i <= m; ++i) {
                cout << ans[i] << '␣';
        }
        cout << '\n';
        //system("pause");
        return 0;
}
```