Олимпиада СПбГУ по информатике 2019/20 учебного года

A	B	C	D	E	F	Sum
100	100	100	40	45	0	385

Task A ()

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    #ifdef LOCAL
    freopen("A.in", "r", stdin);
    #endif // LOCAL
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    int n;
    cin >> n;
    cout << n - 1 << "\n";

    return 0;
}
```

Task B ()

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

using namespace std;

const double ANGLE = 1.0471975511965976;

struct Point {
    double x, y;

    Point(){};

    Point(double x, double y) : x(x), y(y) {};
};

struct Vector {
    double x, y;

    Vector(Point A, Point B) {
        x = B.x - A.x;
        y = B.y - A.y;
    }

    Vector(double x, double y) : x(x), y(y) {};

    double crossProduct(Vector v) {
        return x * v.y - y * v.x;
    }

    double dotProduct(Vector  v) {
        return x * v.x + y * v.y;
    }

    double angle(Vector v) {
        return atan2(crossProduct(v), dotProduct(v));
    }

    Vector rotate(double angle) {
        double c = cos(angle);
        double s = sin(angle);
        double newX = c * x - s * y;
        double newY = s * x + c * y;
        return Vector(newX, newY);
    }
};

void encode() {

    Point p[6];
    for (int i = 0; i < 6; i++) {
        cin >> p[i].x >> p[i].y;
    }
    Point from = p[0];
    int id = 0;
    for (int i = 0; i < 6; i++) {
        if (p[i].x > from.x || (p[i].x == from.x && p[i].y < from.y)) {
            from = p[i];
            id = i;
        }
    }
    #ifdef LOCAL
    //cout << from.x << " " << from.y << endl;
    #endif // LOCAL
    swap(p[id], p[0]);

    auto cmp = [&](Point A, Point B) {
        Vector PA(from, A);
        Vector PB(from, B);
        //cout << PA.x << " " << PA.y << endl;
        //cout << PB.x << " " << PB.y << endl;
        return PA.crossProduct(PB) > 0;
    };
    sort(p, p + 6, cmp);
    cout.precision(15);
    for (int i = 0; i < 2; i++) {
        cout << fixed << p[i].x << "␣" << p[i].y << "\n";
    }
    Point center((p[0].x + p[3].x) / 2.0, (p[0].y + p[3].y) / 2.0);
    cout << fixed << center.x << "␣" << center.y << "\n";
}

Point add(Point A, Vector v) {
    return Point(A.x + v.x, A.y + v.y);
}

void decode() {
    Point A, B;
    cin >> A.x >> A.y >> B.x >> B.y;
    Point center;
    cin >> center.x >> center.y;
    Vector r(center, A);
    for (int i = 0; i < 6; i++) {
        Point cur = add(center, r);
        cout << fixed << cur.x << "␣" << cur.y << "\n";
        r = r.rotate(ANGLE);
    }
}

int main() {
    #ifdef LOCAL
    freopen("A.in", "r", stdin);
    #endif // LOCAL
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    int n;
    cin >> n;
    if (n == 6) {
        encode();
    } else {
        decode();
    }


    return 0;
}
```

Task C ()

```
#include <iostream>
#include <vector>

using namespace std;


vector<int> pF(string s) {
    int n = s.size();
    vector<int> p(n + 1);
    for (int len = 2; len <= n; len++) {
        int k = p[len - 1];
        while (s[k] != s[len - 1] && k > 0) {
            k = p[k];
        }
        if (s[k] == s[len - 1]) {
            k++;
        }
        p[len] = k;
    }
    return p;
}

int solvePref(string &s, string &t) {
    string s1 = s + "#" + t;
    vector<int> p1 = pF(s1);
    int mx = 0;
    for (int i = s.size() + 2; i < p1.size(); i++) {
        mx = max(mx, p1[i]);
    }
    return (int) s.size() - mx;
}

int solve(string &s, string &t) {
    int ans = s.size();
    for (int i = 0; i < s.size(); i++) {
        string cur = s.substr(i);
        ans = min(ans, solvePref(cur, t) + i);
    }
    return ans;
}

const int MAX_N = 505;
const int MAX_M = 1e4 + 10;

int dp[MAX_N][MAX_M];

int solve2(string &s, string &t) {
    int n = s.size();
    int m = t.size();
    for (int i = 0; i < n; i++) {
        dp[i][0] = 0;
    }
    for (int j = 0; j < m; j++) {
        dp[0][j] = 0;
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            dp[i][j] = 0;
            if (s[i - 1] == t[j - 1]) {
                dp[i][j] = max(dp[i][j], dp[i - 1][j - 1] + 1);
            }
            dp[i][j] = max(dp[i][j], dp[i - 1][j]);
        }
    }
    int mx = 0;
    for (int j = 1; j <= m; j++) {
        mx = max(mx, dp[n][j]);
    }
    return n - mx;
}

int main() {
    #ifdef LOCAL
    freopen("A.in", "r", stdin);
    #endif // LOCAL
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    int n;
    string s;
    cin >> s;
    cin >> n;
    int ans = 0;
    for (int i = 0; i < n; i++) {
        string t;
        cin >> t;
        int cur = solve2(s, t);
        ans += cur;
        #ifdef LOCAL
        cout << cur << endl;
        #endif // LOCAL
    }
    cout << ans << "\n";

    return 0;
}
```

Task D ()

```
#include <iostream>
#include <vector>
#include <queue>

using namespace std;

const int MAX_N = 1003;
const int MAX_M = 1003;

int dI[MAX_N][MAX_M];
int dJ[MAX_N][MAX_M];

struct Edge {
    int to;
    int cost;

    Edge(int to, int cost) : to(to), cost(cost){};
};

const int MAX_V = (MAX_N + 1) * (MAX_M + 1);

inline int encode(int i, int j) {
    return i * MAX_N + j;
}

vector<Edge> g[MAX_V];

int cost(int sI, int sJ, int eI, int eJ, int vI, int vJ) {
    int needI = eI - sI;
    int needJ = eJ - sJ;
    int x = needI - vI;
    int y = needJ - vJ;
    return abs(x) + abs(y);
}

int n, m;

inline bool check(int i, int j) {
    return i >= 0 && i < n && j >= 0 && j < m;
}

const int INF = 1e9;

struct Vertex {
    int v, cost;

    Vertex(int v, int cost) : v(v), cost(cost){};

    bool operator >(const Vertex &v) const {
        return cost > v.cost;
    }
};

int dist[MAX_V];
bool used[MAX_V];

int main() {
    #ifdef LOCAL
    freopen("A.in", "r", stdin);
    #endif // LOCAL
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    cin >> n >> m;
    int sI, sJ, eI, eJ;
    cin >> sI >> sJ >> eI >> eJ;
    sI--, sJ--, eI--, eJ--;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> dI[i][j];
            cin >> dJ[i][j];
        }
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            for (int addI = -n; addI <= n; addI++) {
                for (int addJ = -m; addJ <= m; addJ++) {
                    if (addI == 0 && addJ == 0) {
                        continue;
                    }
                    int newI = i + addI;
                    int newJ = j + addJ;
                    if (check(newI, newJ)) {
                        int w = cost(i, j, newI, newJ, dI[i][j], dJ[i][j]);
                        g[encode(i, j)].push_back(Edge(encode(newI, newJ), w));
                    }
                }
            }
        }
    }
    priority_queue<Vertex, vector<Vertex>, greater<Vertex>> q;
    fill(dist, dist + MAX_V, INF);
    dist[encode(sI, sJ)] = 0;
    q.push(Vertex(encode(sI, sJ), 0));
    while (!q.empty()) {
        Vertex ver = q.top();
        q.pop();
        int v = ver.v;
        if (used[v]) {
            continue;
        }
        used[v] = true;
        for (Edge &e : g[v]) {
            int to = e.to;
            int w = e.cost;
            int newW = dist[v] + w;
            if (newW < dist[to]) {
                dist[to] = newW;
                q.push(Vertex(to, newW));
            }
        }
    }
    cout << dist[encode(eI, eJ)] << "\n";

    return 0;
}
```

Task E ()

```
#include <iostream>
#include <vector>
#include <set>
#include <algorithm>
#include <random>
#include <map>

using namespace std;

#define int long long

mt19937 rnd(23);

uniform_int_distribution<int> distr(-1e17, 1e17);

uniform_int_distribution<int> distr2(-1e5, 1e5);



struct Cand {
    int startI, startJ;
    int cnt = 0;

    Cand(int i, int j) : startI(i), startJ(j){};
};

map<pair<int, int>, Cand*> candByCoord;

set<Cand *> active;

const int SHIFT = 1e7;

vector<pair<int, int>> coords;

pair<int, int> next(Cand *c, int step) {
    return {c->startI + coords[step].first, c->startJ + coords[step].second};
}

int cnt = 0;
const int FAKE = -1e17;


pair<int, int> query(vector <pair<int, int>> v) {
    if (v.size() == 1) {
        cnt++;
        v.emplace_back(FAKE + cnt, FAKE + cnt);
    }
    auto gg = v[0];
    cout << "?␣" << gg.first << "␣"<< gg.second << "␣";
    gg = v[1];
    cout << gg.first << "␣"<< gg.second << endl;
    int c, d;
    cin >> c >> d;
    return {c, d};
}

signed main() {
    ios::sync_with_stdio(false);


    int n, m, b;
    cin >> n >> m >> b;
    for (int i = 0; i < b; i++) {
        int x, y;
        cin >> x >> y;
        x--, y--;
        coords.push_back({x, y});
    }
    shuffle(coords.begin(), coords.end(), rnd);
    int curNeed = 1 << b;
    for (int i = 0; i < curNeed; i++) {
        active.insert(new Cand(distr(rnd), distr(rnd)));
    }
    while (true) {
        for (Cand* c: active) {
            if (c->cnt == b) {
                cout << "!␣" << c->startI << "␣" << c->startJ << endl;
                return 0;
            }
        }
        auto it17 = active.begin();
        set<Cand *> newActive;
        while (it17 != active.end()) {
            newActive.insert(*it17);
            it17++;
        }
        auto it = active.begin();
        while (it != active.end()) {
            pair<int, int> gg = next((*it), (*it)->cnt);
            vector <pair<int, int>> v;
            v.push_back(gg);
            candByCoord[gg] = *it;
            (*it)->cnt++;
            it++;
            if (it != active.end()) {
                gg = next(*it, (*it)->cnt);
                v.push_back(gg);
                candByCoord[gg] = *it;
                (*it)->cnt++;
                newActive.insert(*it);
                it++;
            }
            int i, j;
            auto p = query(v);
            i = p.first;
            j = p.second;
            if (candByCoord.count({i, j}) > 0) {
                Cand *x = candByCoord[{i, j}];
                auto it = newActive.find(x);
                if (it != newActive.end()) {
                    newActive.erase(it);
                }
            }
        }
        active = newActive;
    }

    return 0;
}
```

Task F ()