Олимпиада СПбГУ по информатике 2019/20 учебного года

A	B	C	D	E	F	Sum
100	100	100	40	6	0	346

Task A ()

```
#include <iostream>
#include <algorithm>
using namespace std;

signed main() {
    int n;
    cin >> n;
    cout << n - 1;
    return 0;
}
```

Task B ()

```
#include <iostream>
#include <algorithm>
#include <string>
#include <fstream>
#include <vector>
#include <set>
#include <cassert>
using namespace std;

#define double long double
#define Point pair<double, double>
#define x first
#define y second

const int n = 6, INF = 2e9;
const double eps = 1e8;

double operator * (const Point &a, const Point &b) {
    return a.x * b.x + a.y * b.y;
}

double operator ^ (const Point &a, const Point &b) {
    return a.x * b.y - a.y * b.x;
}

Point operator - (const Point &a, const Point &b) {
    return Point(a.x - b.x, a.y - b.y);
}

Point operator + (const Point &a, const Point &b) {
    return Point(a.x + b.x, a.y + b.y);
}

Point operator / (const Point &a, int b) {
    return Point(a.x / b, a.y / b);
}

Point operator * (const Point &a, int b) {
    return Point(a.x * b, a.y * b);
}

Point setter;
bool cmp(const Point &a, const Point &b) {
    Point v1 = a - setter, v2 = b - setter;
    return (v1 ^ v2) > 0;
}

vector<Point> get_sorted(vector<Point> a) {
    int suka = -1;
    setter = Point(INF, INF);
    for (int i = 0; i < n; i++) {
        double q = a[i].x, w = a[i].y;
        if (q < setter.x) {
            setter = Point(q, w);
            suka = i;
        } else if (abs(q - setter.x) < eps) {
            if (w < setter.y) {
                suka = i;
                setter = Point(q, w);
            }
        }
    }
    vector <Point> value;
    value.push_back(setter);
    for (int i = 0; i < n; i++) {
        if (suka == i)
            continue;
        value.push_back(a[i]);
    }
    sort(value.begin() + 1, value.end(), cmp);
    return value;
}
```

```cpp
void encode() {
    vector<Point> a;
    for (int i = 0; i < n; i++) {
        double q, w;
        cin >> q >> w;
        a.push_back(Point(q, w));
    }
    vector<Point> value = get_sorted(a);
    vector<Point> send(3);
    send[0] = (value[0] + value[1]) / 2;
    send[1] = (value[3] + value[4]) / 2;
    send[2] = value[5];
    for (int i = 0; i < send.size(); i++)
        cout << send[i].x << "␣" << send[i].y << '\n';
}

void decode() {
    vector<Point> data, ans;
    for (int i = 0; i < n / 2; i++) {
        double q, w;
        cin >> q >> w;
        data.push_back(Point(q, w));
    }
    ans.push_back(data[2]);
    Point center = (data[0] + data[1]) / 2;
    Point v1 = center - data[2];
    ans.push_back((center + v1));
    Point v2 = v1 / 2;
    ans.push_back(data[0] + v2);
    ans.push_back(data[0] - v2);
    ans.push_back(data[1] + v2);
    ans.push_back(data[1] - v2);
    ans = get_sorted(ans);
    for (int i = 0; i < n; i++)
        cout << ans[i].x << "␣" << ans[i].y << '\n';
}

void run() {
    //freopen("input.txt", "r", stdin);
    int trash;
    cin >> trash;
    if (trash == 6) {
        encode();
    } else {
        decode();
    }
}


signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cout.setf(ios::fixed);
    cout.precision(10);
    run();
    return 0;
}
```

Task C ()

```cpp
#include <iostream>
#include <algorithm>
#include <string>
#include <fstream>
using namespace std;

signed main() {
    //freopen("input.txt", "r", stdin);
    string pattern;
    cin >> pattern;
    int n;
    cin >> n;
    long long ans = 0;
    for (int i = 0; i < n; i++) {
        string s;
        cin >> s;
        int temp = 2e9;
        for (int j = 0; j < s.size(); j++) {
            int mn = 0, pos_a = j, pos_b = 0;
            for (int pos_b = 0; pos_b < pattern.size(); pos_b++) {
                if (pos_a >= s.size()) {
                    mn++;
                } else {
                    if (s[pos_a] == pattern[pos_b]) {
                        pos_a++;
                    } else {
                        mn++;
                    }
                }
            }
            temp = min(temp, mn);
        }
        //cout << temp << '\n';
        ans += temp;
    }
    cout << ans;
    return 0;
}
```

Task D ()

```
#include <iostream>
#include <algorithm>
#include <string>
#include <fstream>
#include <vector>
#include <set>
#include <cassert>
using namespace std;

const int MAXN = 1e3 + 10, INF = 2e9;

pair<int, int> field[MAXN][MAXN], a, b;
vector<pair<pair<int, int>, int> > graph[MAXN][MAXN];
int dist[MAXN][MAXN];
bool used[MAXN][MAXN];

bool relax(int &a, int b) {
    if (b < a) {
        a = b;
        return true;
    }
    return false;
}

void run() {
    //freopen("input.txt", "r", stdin);
    int n, m;
    cin >> n >> m;
    cin >> a.first >> a.second >> b.first >> b.second;
    a.first--;
    a.second--;
    b.first--;
    b.second--;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> field[i][j].first >> field[i][j].second;
        }
    }
    const int MAXD = 15;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            for (int di = -MAXD; di <= MAXD; di++) {
                for (int dj = -MAXD; dj <= MAXD; dj++) {
                    if (!(i + di >= 0 && i + di < n && j + dj >= 0 && j + dj < m))
                        continue;
                    if (abs(di) + abs(dj) > MAXD)
                        continue;
                    auto u = abs(field[i][j].first - di);
                    auto v = abs(field[i][j].second - dj);
                    graph[i][j].push_back(make_pair(make_pair(i + di, j + dj), u + v));
                }
            }
        }
    }
    fill(&dist[0][0], &dist[0][0] + MAXN * MAXN, INF);
    dist[a.first][a.second] = 0;
    set<pair<int, pair<int, int> > > order;
    order.insert(make_pair(dist[a.first][a.second], make_pair(a.first, a.second)));
    while (!order.empty()) {
        while (true) {
            if (order.empty())
                break;
            pair<int, int> top = (*order.begin()).second;
            if (used[top.first][top.second]) {
                order.erase(order.begin());
            } else {
                break;
            }
        }
        if (order.empty())
```

```
            break;
        pair<int, pair<int, int> > top = (*order.begin());
        auto i = top.second.first, j = top.second.second;
        order.erase(order.begin());
        used[i][j] = true;
        for (int id = 0; id < graph[i][j].size(); id++) {
            int nxt_i = graph[i][j][id].first.first;
            int nxt_j = graph[i][j][id].first.second;
            int cost = graph[i][j][id].second;
            if (used[nxt_i][nxt_j])
                continue;
            if (relax(dist[nxt_i][nxt_j], dist[i][j] + cost)) {
                order.insert(make_pair(dist[nxt_i][nxt_j], make_pair(nxt_i, nxt_j)));
            }
        }
    }
    assert(dist[b.first][b.second] != INF);
    cout << dist[b.first][b.second];
}


signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    run();
    return 0;
}
```

Task E ()

```
#include <iostream>
#include <algorithm>
#include <string>
#include <fstream>
#include <vector>
#include <set>
#include <cassert>
using namespace std;

int n, m, b;

void send_ans(pair<int, int> a) {
    cout << "!␣" << a.first << "␣" << a.second << endl;
}

pair<int, int> ask(pair<int, int> a, pair<int, int> b) {
    cout << "?␣" << a.first << "␣" << a.second << "␣" << b.first << "␣" << b.second << endl;
    pair<int, int> fuck;
    cin >> fuck.first >> fuck.second;
    return fuck;
}

int manhatan(const pair<int, int> &a, const pair<int, int> &b) {
    return abs(a.first - b.first) + abs(a.second - b.second);
}

bool FIRST_GROUP() {
    if (b != 1)
        return false;
    pair<int, int> pos;
    cin >> pos.first >> pos.second;
    pair<int, int> a = make_pair((int)-1e5, 0), b = make_pair((int)1e5, 0);
    pair<int, int> reply = ask(a, b);
    int di, dj;
    if (manhatan(a, reply) < manhatan(b, reply)) {
        di = b.first - pos.first;
        dj = b.second - pos.second;
    } else {
        di = a.first - pos.first;
        dj = a.second - pos.second;
    }
    pair<int, int> ans = make_pair(1 + di, 1 + dj);
    send_ans(ans);
    return true;
}

void run() {
    //freopen("input.txt", "r", stdin);
    cin >> n >> m >> b;
    if (FIRST_GROUP())
        return;
    assert(false);
}


signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cout.setf(ios::fixed);
    cout.precision(10);
    run();
    return 0;
}
```

Task F ()