Олимпиада СПбГУ по информатике 2019/20 учебного года

A	B	C	D	E	F	Sum
100	100	100	40	0	0	340

Task A ()

```cpp
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

int main()
{
    //freopen("input.in", "r", stdin);
    //freopen("input.out", "w", stdout);
    ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int n;
    cin >> n;
    cout << n - 1;
    return 0;
}

/**
**/
```

Task B ()

```
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;
typedef long double ld;

const ld pi = 3.14, EPS = 1e-2;

struct vect {
    ld x, y;
};

struct point {
    ld x, y;
    void print() {
        cout.precision(15);
        cout << fixed << x << " " << y << "\n";
    }
};

vect get_vect(point a, point b) {
        vect v = {b.x - a.x, b.y - a.y};
        return v;
}

point get_point(point P, vect V) {
    return {P.x + V.x, P.y + V.y};
}

ld dist(point a, point b) {
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}

bool check(vector<point>& P) {
    ld d = dist(P[0], P[1]);
    for (int i = 0; i < 6; ++i) {
        //cout << d << " " << dist(P[i], P[(i + 1) % 6]) << endl;
        if (abs(dist(P[i], P[(i + 1) % 6]) - d) > EPS)
            return 0;
    }
    return 1;
}

void make_ok(vector<point>& A) {
    vector<int> p = {0, 1, 2, 3, 4, 5};
    vector<point> cur = A;
    while (true) {
        for (int i = 0; i < 6; ++i)
            cur[i] = A[p[i]];
        if (check(cur)) {
            A = cur;
            return;
        }
        if (!next_permutation(p.begin(), p.end()))
            exit(0);
    }
}

void solve1() {
    vector<point> P;
    /*P = {{10.000, 0.000},
        {5.000, 8.660},
        {-5.000, 8.660},
        {-10.000, -0.000},
        {-5.000, -8.660},
        {5.000, -8.660}};*/
    for (int i = 0; i < 6; ++i) {
        ld x, y;
        cin >> x >> y;
        //cout << x << " " << y << endl;
        P.push_back({x, y});
```

```
    }
    make_ok(P);
    point a = P[0], b = P[3];
    vect v = {(b.x - a.x) / 2, (b.y - a.y) / 2};
    point c = get_point(a, v);
    P[1].print();
    c.print();
    P[2].print();
}

void solve2() {
    vector<point> T;
    for (int i = 0; i < 3; ++i) {
        ld x, y;
        cin >> x >> y;
        T.push_back({x, y});
    }
    vect v1 = get_vect(T[0], T[1]);
    vect v2 = get_vect(T[2], T[1]);
    vect v3 = get_vect(T[0], T[2]);
    vect v4 = get_vect(T[2], T[0]);
    vector<point> P = {T[0], T[2], get_point(T[1], v1), get_point(T[1], v2), get_point(T[1], v3),
        get_point(T[1], v4)};
    make_ok(P);
    for (int i = 0; i < 6; ++i) {
        P[i].print();
    }
}

int main()
{
    //freopen("input.txt", "r", stdin);
    //freopen("output.txt", "w", stdout);
    //ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int n;
    cin >> n;
    if (n == 6) {
        solve1();
    } else {
        solve2();
    }
    return 0;
}

/**
**/
```

Task C ()

```cpp
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

int m;
string t;

const int inf = 1e6;

int calc() {
    string s;
    cin >> s;
    int n = s.size();
    vector<vector<vector<int>>> dp(n + 5, vector<vector<int>>(m + 5, vector<int>(2, inf)));
    for (int i = 0; i < n; ++i) {
        dp[i][m - 1][0] = 1;
        if (s[i] == t[m - 1])
            dp[i][m - 1][1] = 0;
    }
    for (int j = 0; j < m; ++j) {
        dp[n - 1][j][0] = m - j;
        if (s[n - 1] == t[j])
            dp[n - 1][j][1] = m - j - 1;
    }
    int ans = m;
    for (int j = m - 2; j >= 0; --j) {
        for (int i = 0; i < n - 1; ++i) {
            dp[i][j][0] = min({dp[i][j + 1][0] + 1, dp[i + 1][j + 1][1] + 1});
            if (s[i] == t[j])
                dp[i][j][1] = min({dp[i][j + 1][0], dp[i + 1][j + 1][1]});
        }
    }
    for (int i = 0; i < n; ++i)
        ans = min({ans, dp[i][0][0], dp[i][0][1]});
    for (int j = 0; j < m; ++j) {
        ans = min(ans, dp[0][j][1] + j);
    }
    return ans;
}

int main()
{
    //freopen("input.in", "r", stdin);
    //freopen("input.out", "w", stdout);
    //ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    cin >> t;
    m = t.size();
    int ans = 0;
    int k;
    cin >> k;
    while (k--) {
        int tmp = calc();
        //cout << tmp << endl;
        ans += tmp;

    }
    cout << ans;
    return 0;
}

/**
**/
```

Task D ()

```cpp
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

struct vect {
    int x, y;
};

const int N = 1003;

int n, m;
vector<vector<int>> C(N, vector<int>(N, -1));
vector<vector<int>> dist(N, vector<int>(N, N * N));
vector<vector<bool>> used(N, vector<bool>(N, 0));
vector<vector<vect>> V(N, vector<vect>(N));
int cur = 0;
int sx, sy, fx, fy;

bool ok(int x, int y) {
    return 0 <= x && x < n && 0 <= y && y < m;
}

void dijkstra() {
    dist[sx][sy] = 0;
    set<pair<int, pair<int, int>>> S = {make_pair(0, make_pair(sx, sy))};
    while (!S.empty()) {
        pair<int, int> tmp = (*S.begin()).second;
        S.erase(S.begin());
        int x = tmp.first, y = tmp.second;
        //cout << x << " " << y << " " << dist[x][y] << endl;
        used[x][y] = 1;
        for (int tx = 0; tx < n; ++tx) {
            for (int ty = 0; ty < m; ++ty) {
                if (!used[tx][ty]) {
                    S.erase(make_pair(dist[tx][ty], make_pair(tx, ty)));
                    dist[tx][ty] = min(dist[tx][ty], dist[x][y] + abs(tx - (x + V[x][y].x)) + abs(
                        ty - (y + V[x][y].y)));
                    S.insert(make_pair(dist[tx][ty], make_pair(tx, ty)));
                }
            }
        }
    }
}

int main()
{
    //freopen("input.text", "r", stdin);
    //ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    cin >> n >> m;
    cin >> sx >> sy >> fx >> fy;
    --sx, --sy, --fx, --fy;
    //cout << n << " " << m << endl;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            int r, c;
            cin >> r >> c;
            V[i][j] = {r, c};
        }
    }
    /*V[0][0] = {0, 1};
    V[0][1] = {1, 0};
    V[0][2] = {0, -1};
    V[1][0] = {-1, -1};
    V[1][1] = {1, 0};
    V[1][2] = {-1, 0};
    V[2][0] = {0, 0};
    V[2][1] = {0, -1};
    V[2][2] = {-1, 0};*/
    dijkstra();
    cout << dist[fx][fy];
```

```
    return 0;
}

/**
**/
```

Task E ()

Task F ()