Олимпиада СПбГУ по информатике 2019/20 учебного года

A	B	C	D	E	F	Sum
100	100	100	40	100	0	440

Task A ()

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

using namespace std;

#define int long long

vector<int> z_func(string s)
{
    int n = s.size();
    vector<int> pi(n);
    for (int i = 1, l = 0, r = 0; i < n; ++i)
    {
        cout << r << endl;
        if (i <= r)
        {
            pi[i] = min(r - i + 1, pi[i - l]);
        }
        while (i + pi[i] < n && s[i + pi[i]] == s[pi[i]])
        {
            pi[i]++;
        }
        if (i + pi[i] > r)
        {
            r = i + pi[i] - 1;
            l = i;
        }
    }
    return pi;
}

vector<int> tree;

int get(int v, int tl, int tr, int l, int r)
{
    if (l == tl && r == tr)
    {
        return tree[v];
    }
    if (tl >= r)
    {
        return 0;
    }
    if (l >= tr)
    {
        return 0;
    }
    if (l >= r)
    {
        return 0;
    }
    int tm = (tl + tr) / 2;
    return get(2 * v + 1, tl, tm, l, min(tm, r)) + get(2 * v + 2, tm, tr, max(tm, l), r);
}


void upd(int v, int tl, int tr, int pos, int val)
```

```
{
    if (pos >= tr || pos < tl)
    {
        return;
    }
    if (tl == tr - 1)
    {
        tree[v] = val;
        return;
    }
    int tm = (tl + tr) / 2;
    upd(2 * v + 1, tl, tm, pos, val);
    upd(2 * v + 2, tm, tr, pos, val);
    tree[v] = tree[2 * v + 1] + tree[2 * v + 2];
}

vector<int> used;
vector<vector<int> > graph;

void dfs(int v)
{
    if (used[v])
    {
        return;
    }
    for (int i = 0; i < graph[v].size(); ++i)
    {
        dfs(graph[v][i]);
    }
}


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

Task B ()

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

using namespace std;

#define int long long

vector<int> z_func(string s)
{
    int n = s.size();
    vector<int> pi(n);
    for (int i = 1, l = 0, r = 0; i < n; ++i)
    {
        cout << r << endl;
        if (i <= r)
        {
            pi[i] = min(r - i + 1, pi[i - l]);
        }
        while (i + pi[i] < n && s[i + pi[i]] == s[pi[i]])
        {
            pi[i]++;
        }
        if (i + pi[i] > r)
        {
            r = i + pi[i] - 1;
            l = i;
        }
    }
    return pi;
}

vector<int> tree;

int get(int v, int tl, int tr, int l, int r)
{
    if (l == tl && r == tr)
    {
        return tree[v];
    }
    if (tl >= r)
    {
        return 0;
    }
    if (l >= tr)
    {
        return 0;
    }
    if (l >= r)
    {
        return 0;
    }
    int tm = (tl + tr) / 2;
    return get(2 * v + 1, tl, tm, l, min(tm, r)) + get(2 * v + 2, tm, tr, max(tm, l), r);
}


void upd(int v, int tl, int tr, int pos, int val)
{
    if (pos >= tr || pos < tl)
    {
        return;
    }
    if (tl == tr - 1)
    {
        tree[v] = val;
        return;
    }
    int tm = (tl + tr) / 2;
    upd(2 * v + 1, tl, tm, pos, val);
    upd(2 * v + 2, tm, tr, pos, val);
    tree[v] = tree[2 * v + 1] + tree[2 * v + 2];
}
```

```
vector<int> used;
vector<vector<int> > graph;

void dfs(int v)
{
    if (used[v])
    {
        return;
    }
    for (int i = 0; i < graph[v].size(); ++i)
    {
        dfs(graph[v][i]);
    }
}


struct Point
{
    long double x, y;
};

long double vp(long double x1, long double y1, long double x2, long double y2)
{
    return x1 * y2 - x2 * y1;
}

signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    int n;
    cin >> n;
    if (n == 6)
    {
        vector<long double> x(n);
        vector<long double> y(n);
        for (int i = 0; i < n; ++i)
        {
            cin >> x[i];
            cin >> y[i];
        }
        long double min1 = 1e9;
        for (int i = 0; i < n; ++i)
        {
            for (int j = i + 1; j < n; ++j)
            {
                min1 = min(min1, sqrt((x[i] - x[j]) * (x[i] - x[j]) + (y[i] - y[j]) * (y[i] - y[j
                    ])));
            }
        }
        vector<pair<long double, long double> > result;
        result.push_back({x[0], y[0]});
        for (int i = 1; i < n; ++i)
        {
            if (sqrt((x[i] - x[0]) * (x[i] - x[0]) + (y[i] - y[0]) * (y[i] - y[0])) > min1 - 0.01
                && sqrt((x[i] - x[0]) * (x[i] - x[0]) + (y[i] - y[0]) * (y[i] - y[0])) < min1 +
                0.01)
            {
                result.push_back({x[i], y[i]});
            }
        }
        for (int i = 0; i < result.size(); ++i)
        {
            cout << result[i].first << ' ' << result[i].second << endl;
            //printf("%f.3 %f.3\n", result[i].first , result[i].second);
        }
    }
    else
    {
        //return 1;
        vector<pair<long double, long double> > result(3);
        for (int i = 0; i < result.size(); ++i)
        {
```

```
            cin >> result[i].first >> result[i].second;
        }
        Point vect1;
        vect1.x = result[0].first - result[1].first;
        vect1.y = result[0].second - result[1].second;
        Point vect2;
        vect2.x = result[2].first - result[1].first;
        vect2.y = result[2].second - result[1].second;
        long double x4 = result[2].first - vect2.y / sqrt(3);
        long double y4 = result[2].second + vect2.x / sqrt(3);
        long double x3 = result[2].first + vect2.y / sqrt(3);
        long double y3 = result[2].second - vect2.x / sqrt(3);
        if (vp(vect1.x, vect1.y, vect2.x, vect2.y) * vp(x3 - result[1].first, y3 - result[1].
            second, vect2.x, vect2.y) <= 0)
        {
            x4 = result[1].first + vect2.y / sqrt(3);
            y4 = result[1].second - vect2.x / sqrt(3);
            long double x5 = result[0].first + 2 * vect2.y / sqrt(3);
            long double y5 = result[0].second - 2 * vect2.x / sqrt(3);
            /*printf("%f.3 %f.3\n", result[1].first, result[1].second);
            printf("%f.3 %f.3\n", result[0].first, result[0].second);
            printf("%f.3 %f.3\n", result[2].first, result[2].second);
            printf("%f.3 %f.3\n", x3, y3);
            printf("%f.3 %f.3\n", x5, y5);
            printf("%f.3 %f.3\n", result[i].first, result[i].second);*/
            cout << result[1].first << ' ' << result[1].second << endl;

            cout << result[0].first << ' ' << result[0].second << endl;

            cout << result[2].first << ' ' << result[2].second << endl;

            cout << x3 << ' ' << y3 << endl;

            cout << x5 << ' ' << y5 << endl;

            cout << x4 << ' ' << y4 << endl;
        }
        else
        {

            x3 = result[1].first - vect2.y / sqrt(3);
            y3 = result[1].second + vect2.x / sqrt(3);
            long double x5 = result[0].first - 2 * vect2.y / sqrt(3);
            long double y5 = result[0].second + 2 * vect2.x / sqrt(3);
            cout << result[1].first << ' ' << result[1].second << endl;

            cout << result[0].first << ' ' << result[0].second << endl;

            cout << result[2].first << ' ' << result[2].second << endl;

            cout << x4 << ' ' << y4 << endl;

            cout << x5 << ' ' << y5 << endl;

            cout << x3 << ' ' << y3 << endl;
        }
    }
}


//6 10.000 0.000 5.000 8.660 -5.000 8.660 -10.000 -0.000 -5.000 -8.660 5.000 -8.660
```

Task C ()

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

using namespace std;

#define int long long

vector<int> z_func(string s)
{
    int n = s.size();
    vector<int> pi(n);
    for (int i = 1, l = 0, r = 0; i < n; ++i)
    {
        cout << r << endl;
        if (i <= r)
        {
            pi[i] = min(r - i + 1, pi[i - l]);
        }
        while (i + pi[i] < n && s[i + pi[i]] == s[pi[i]])
        {
            pi[i]++;
        }
        if (i + pi[i] > r)
        {
            r = i + pi[i] - 1;
            l = i;
        }
    }
    return pi;
}

signed main()
{
    string s;
    cin >> s;
    int n;
    cin >> n;
    int cnt = 0;
    for (int i = 0; i < n; ++i)
    {
        string s2;
        cin >> s2;
        int min1 = 1e9;
        for (int j = 0; j < s2.size(); ++j)
        {
            int tmpi = 0;
            int tmpj = j;
            int cnt2 = 0;
            while (tmpi < s.size())
            {
                if (tmpj == s2.size() || s2[tmpj] != s[tmpi])
                {
                    tmpi++;
                    cnt2++;
                }
                else
                {
                    tmpi++;
                    tmpj++;
                }
            }
            min1 = min(min1, cnt2);
        }
        cnt += min1;
    }
    cout << cnt;
}
```

Task D ()

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

using namespace std;

#define int long long

vector<int> z_func(string s)
{
    int n = s.size();
    vector<int> pi(n);
    for (int i = 1, l = 0, r = 0; i < n; ++i)
    {
        cout << r << endl;
        if (i <= r)
        {
            pi[i] = min(r - i + 1, pi[i - l]);
        }
        while (i + pi[i] < n && s[i + pi[i]] == s[pi[i]])
        {
            pi[i]++;
        }
        if (i + pi[i] > r)
        {
            r = i + pi[i] - 1;
            l = i;
        }
    }
    return pi;
}

vector<int> tree;

int get(int v, int tl, int tr, int l, int r)
{
    if (l == tl && r == tr)
    {
        return tree[v];
    }
    if (tl >= r)
    {
        return 0;
    }
    if (l >= tr)
    {
        return 0;
    }
    if (l >= r)
    {
        return 0;
    }
    int tm = (tl + tr) / 2;
    return get(2 * v + 1, tl, tm, l, min(tm, r)) + get(2 * v + 2, tm, tr, max(tm, l), r);
}


void upd(int v, int tl, int tr, int pos, int val)
{
    if (pos >= tr || pos < tl)
    {
        return;
    }
    if (tl == tr - 1)
    {
        tree[v] = val;
        return;
    }
    int tm = (tl + tr) / 2;
    upd(2 * v + 1, tl, tm, pos, val);
    upd(2 * v + 2, tm, tr, pos, val);
    tree[v] = tree[2 * v + 1] + tree[2 * v + 2];
}
```

```
vector<int> used;
vector<vector<int> > graph;

void dfs(int v)
{
    if (used[v])
    {
        return;
    }
    for (int i = 0; i < graph[v].size(); ++i)
    {
        dfs(graph[v][i]);
    }
}


signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    int n, m;
    cin >> n >> m;
    int x1, y1, x2, y2;
    cin >> x1 >> y1 >> x2 >> y2;
    x1--;
    y1--;
    x2--;
    y2--;
    vector<vector<int> > field1(n, vector<int> (m));
    vector<vector<int> > field2(n, vector<int> (m));
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < m; ++j)
        {
            cin >> field1[i][j] >> field2[i][j];
        }
    }
    set<pair<int, pair<int, int> > > s;
    vector<vector<int> > result(n, vector<int> (m, 1e9));
    s.insert({0, {x1, y1}});
    while (!s.empty())
    {
        int l = (*s.begin()).first;
        int x = (*s.begin()).second.first;
        int y = (*s.begin()).second.second;
        //cout << x << ' ' << y << endl;
        result[x][y] = l;
        s.erase(s.begin());
        for (int i = 0; i < n; ++i)
        {
            for (int j = 0; j < m; ++j)
            {
                if (result[i][j] > result[x][y] + abs(i - (x + field1[x][y])) + abs(j - (y + field2
                    [x][y])))
                {
                    s.erase({result[i][j], {i, j}});
                    result[i][j] = result[x][y] + abs(i - (x + field1[x][y])) + abs(j - (y + field2
                        [x][y]));
                    s.insert({result[i][j], {i, j}});
                }
            }
        }
    }
    cout << result[x2][y2];
}

//3 3 1 1 3 3 0 1 1 0 0 -1 -1 -1 1 0 -1 0 0 0 0 -1 -1 0
```

Task E ()

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

using namespace std;

#define int long long

vector<int> z_func(string s)
{
    int n = s.size();
    vector<int> pi(n);
    for (int i = 1, l = 0, r = 0; i < n; ++i)
    {
        cout << r << endl;
        if (i <= r)
        {
            pi[i] = min(r - i + 1, pi[i - l]);
        }
        while (i + pi[i] < n && s[i + pi[i]] == s[pi[i]])
        {
            pi[i]++;
        }
        if (i + pi[i] > r)
        {
            r = i + pi[i] - 1;
            l = i;
        }
    }
    return pi;
}

vector<int> tree;

int get(int v, int tl, int tr, int l, int r)
{
    if (l == tl && r == tr)
    {
        return tree[v];
    }
    if (tl >= r)
    {
        return 0;
    }
    if (l >= tr)
    {
        return 0;
    }
    if (l >= r)
    {
        return 0;
    }
    int tm = (tl + tr) / 2;
    return get(2 * v + 1, tl, tm, l, min(tm, r)) + get(2 * v + 2, tm, tr, max(tm, l), r);
}


void upd(int v, int tl, int tr, int pos, int val)
{
    if (pos >= tr || pos < tl)
    {
        return;
    }
    if (tl == tr - 1)
    {
        tree[v] = val;
        return;
    }
    int tm = (tl + tr) / 2;
    upd(2 * v + 1, tl, tm, pos, val);
    upd(2 * v + 2, tm, tr, pos, val);
    tree[v] = tree[2 * v + 1] + tree[2 * v + 2];
}
```

```
signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    int n, m, b;
    cin >> n >> m >> b;
    vector<int> px(b);
    vector<int> py(b);
    for (int i = 0; i < b; ++i)
    {
        cin >> px[i];
        px[i]--;
        cin >> py[i];
        py[i]--;
    }
    vector<int> used(1 << b);
    for (int h = 0; h < b; ++h)
    {
        int i = 0;
        int j = 0;
        int cnt = 0;
        while (i < (1 << b) && cnt < (1 << (b - h)))
        {
            while (used[i])
            {
                i++;
            }
            j = i + 1;
            while (used[j])
            {
                j++;
            }
            cout << "? " << (i * n + px[h]) << " " << py[h] << " " << (j * n + px[h]) << " " << py
                [h] << endl;
            int x, y;
            cin >> x >> y;
            x /= n;

            used[x] = 1;
            cnt += 2;
            i = j + 1;
        }
    }
    for (int i = 0; i < (1 << b); ++i)
    {
        if (!used[i])
        {
            cout << "! " << i * n << ' ' << 0 << endl;
            return 0;
        }
    }
}

//3 3 1 1 3 3 0 1 1 0 0 -1 -1 -1 1 0 -1 0 0 0 0 -1 -1 0
```

Task F ()