Олимпиада СПбГУ по информатике 2019/20 учебного года

A	B	C	D	E	F	Sum
100	100	100	40	12	5	357

Task A ()

```
//#pragma GCC optimize("Ofast,no-stack-protector,unroll-loops,fast-math")
//#pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,avx,avx2,tune=native")

#include <bits/stdc++.h>

using namespace std;

#define ll long long
#define ld long double
#define pll pair<ll, ll>

ll const N = 2e5 + 1, M = 228228227, P = 257;
ld const EPS = 1e-10;

void FastIO() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
}

struct Node {
    ll x, sz, pr;
    Node* left = nullptr;
    Node* right = nullptr;
    Node() {}
    Node(ll _x) {
        ll x = _x;
        sz = 1;
        pr = rand();
    }
};

ll GetSz(Node* tree) {
    return (tree ? tree->sz : 0);
}

void Upd(Node* tree) {
    tree->sz = GetSz(tree->left) + GetSz(tree->right) + 1;
}

Node* Merge(Node* left, Node* right) {
    if (! left) {
        return right;
    }
    if (! right) {
        return left;
    }

    if (left->pr > right->pr) {
        left->right = Merge(left->right, right);
        Upd(left);
        return left;
    }
    else {
        right->left = Merge(left, right->left);
        Upd(right);
```

```
        return right;
    }
}

pair<Node*, Node*> Split(Node* tree, ll x) {
    if (! tree) {
        return {tree, tree};
    }

    if (GetSz(tree->left) == x - 1) {
        auto tmp = tree->right;
        tree->right = nullptr;
        Upd(tree);
        return {tree, tmp};
    }
    else if (GetSz(tree->left) > x - 1) {
        auto tmp = Split(tree->left, x);
        tree->left = tmp.second;
        Upd(tree);
        return {tmp.first, tree};
    }
    else {
        auto tmp = Split(tree->right, x - GetSz(tree->left) - 1);
        tree->right = tmp.first;
        Upd(tree);
        return {tree, tmp.second};
    }
}

Node* Ins(Node* tree, ll x, ll pos) {
    Node* node = new Node(x);
    auto tmp = Split(tree, pos - 1);
    return Merge(tmp.first, Merge(node, tmp.second));
}

void Zfunc(string s) {
    ll n = s.size();
    vector<ll> z(n, 0);
    ll l = 0, r = 0;
    for (int i = 1; i < n; ++i) {
        if (r >= i) {
            z[i] = min(r - i + 1, z[i - l]);
        }
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
            ++z[i];
        }
        if (i + z[i] - 1 > r) {
            r = i + z[i] - 1;
            l = i;
        }
    }
}

ll bn(ll x, ll n, ll m) {
    if (n == 0) {
        return 1;
    }

    if (n % 2) {
        return (bn(x, n - 1, m) * x) % m;
    }
    else {
        ll tmp = bn(x, n / 2, m);
        return (tmp * tmp) % m;
    }
}

ll gcd(ll a, ll b) {
    while (a && b) {
        if (a > b) {
            a = a % b;
        }
        else {
            b = b % a;
```

```
            }
        }
        return a + b;
}

signed main() {
    FastIO();

    srand(time(NULL));

    ll n;
    cin >> n;

    cout << n - 1;

    return 0;
}
```

Task B ()

```
//#pragma GCC optimize("Ofast,no-stack-protector,unroll-loops,fast-math")
//#pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,avx,avx2,tune=native")

#include <bits/stdc++.h>

using namespace std;

#define ll long long
#define ld long double
#define pll pair<ll, ll>
#define point pair<ld, ld>

ll const N = 2e5 + 1, M = 228228227, P = 257;
ld const EPS = 1e-10;

void FastIO() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
}

struct Node {
    ll x, sz, pr;
    Node* left = nullptr;
    Node* right = nullptr;
    Node() {}
    Node(ll _x) {
        ll x = _x;
        sz = 1;
        pr = rand();
    }
};

ll GetSz(Node* tree) {
    return (tree ? tree->sz : 0);
}

void Upd(Node* tree) {
    tree->sz = GetSz(tree->left) + GetSz(tree->right) + 1;
}

Node* Merge(Node* left, Node* right) {
    if (! left) {
        return right;
    }
    if (! right) {
        return left;
    }

    if (left->pr > right->pr) {
        left->right = Merge(left->right, right);
        Upd(left);
        return left;
    }
    else {
        right->left = Merge(left, right->left);
        Upd(right);
        return right;
    }
}

pair<Node*, Node*> Split(Node* tree, ll x) {
    if (! tree) {
        return {tree, tree};
    }

    if (GetSz(tree->left) == x - 1) {
        auto tmp = tree->right;
        tree->right = nullptr;
        Upd(tree);
        return {tree, tmp};
    }
```

```
    else if (GetSz(tree->left) > x - 1) {
        auto tmp = Split(tree->left, x);
        tree->left = tmp.second;
        Upd(tree);
        return {tmp.first, tree};
    }
    else {
        auto tmp = Split(tree->right, x - GetSz(tree->left) - 1);
        tree->right = tmp.first;
        Upd(tree);
        return {tree, tmp.second};
    }
}

Node* Ins(Node* tree, ll x, ll pos) {
    Node* node = new Node(x);
    auto tmp = Split(tree, pos - 1);
    return Merge(tmp.first, Merge(node, tmp.second));
}

void Zfunc(string s) {
    ll n = s.size();
    vector<ll> z(n, 0);
    ll l = 0, r = 0;
    for (int i = 1; i < n; ++i) {
        if (r >= i) {
            z[i] = min(r - i + 1, z[i - l]);
        }
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
            ++z[i];
        }
        if (i + z[i] - 1 > r) {
            r = i + z[i] - 1;
            l = i;
        }
    }
}

ll bn(ll x, ll n, ll m) {
    if (n == 0) {
        return 1;
    }

    if (n % 2) {
        return (bn(x, n - 1, m) * x) % m;
    }
    else {
        ll tmp = bn(x, n / 2, m);
        return (tmp * tmp) % m;
    }
}

ll gcd(ll a, ll b) {
    while (a && b) {
        if (a > b) {
            a = a % b;
        }
        else {
            b = b % a;
        }
    }
    return a + b;
}

ld kos(point a, point b, point c) {
    return ((a.first - c.first) * (b.second - c.second) - (b.first - c.first) * (a.second - c.
        second));
}

signed main() {
    FastIO();

    srand(time(NULL));
```

```
cout << setprecision(10) << fixed;

//freopen("input.txt", "r", stdin);

ll n;
cin >> n;
if (n == 6) {
    vector<point> v;
    for (int i = 0; i < n; ++i) {
        ld x, y;
        cin >> x >> y;
        v.push_back({x, y});
    }
    sort(v.begin(), v.end());
    set<point> s;
    for (int i = 0; i < n; ++i) {
        s.insert(v[i]);
    }

    vector<point> hull;
    hull.push_back(v[0]);
    s.erase(v[0]);
    for (int i = 0; i < n - 1; ++i) {
        set<point> cp;
        cp = s;
        point cur = *cp.begin();
        cp.erase(cur);
        ll sz = cp.size();
        for (int j = 0; j < sz; ++j) {
            point p = *cp.begin();
            cp.erase(p);
            if (kos(cur, p, hull.back()) < 0) {
                cur = p;
            }
        }
        s.erase(cur);
        hull.push_back(cur);
    }

    cout << hull[0].first << ' ' << hull[0].second << '\n';
    cout << hull[2].first << ' ' << hull[2].second << '\n';
    cout << hull[4].first << ' ' << hull[4].second << '\n';
}
else {
    vector<point> v;
    for (int i = 0; i < n; ++i) {
        ld x, y;
        cin >> x >> y;
        v.push_back({x, y});
    }
    v.push_back(v[0]);
    v.push_back(v[1]);
    ld d = sqrt((v[0].first - v[1].first) * (v[0].first - v[1].first) + (v[0].second - v[1].
        second) * (v[0].second - v[1].second));
    ld f = d / (2 * sqrt(3));
    for (int i = 0; i < n; ++i) {
        cout << v[i].first << ' ' << v[i].second << '\n';
        point mid = {(v[i].first + v[i + 1].first) / 2, (v[i].second + v[i + 1].second) / 2};
        point p = v[i + 1];
        ld a = mid.second - p.second;
        ld b = p.first - mid.first;
        ld na = a / sqrt(a * a + b * b);
        ld nb = b / sqrt(a * a + b * b);
        point o = {mid.first + na * f, mid.second + nb * f};
        point k = {mid.first - na * f, mid.second - nb * f};
        if (kos(o, mid, v[i]) * kos(v[i + 2], mid, v[i]) < 0) {
            cout << o.first << ' ' << o.second << '\n';
        }
        else {
            cout << k.first << ' ' << k.second << '\n';
        }
    }
}
```

```
    return 0;
}
```

Task C ()

```
//#pragma GCC optimize("Ofast,no-stack-protector,unroll-loops,fast-math")
//#pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,avx,avx2,tune=native")

#include <bits/stdc++.h>

using namespace std;

#define ll long long
#define ld long double
#define pll pair<ll, ll>

ll const N = 2e5 + 1, M = 228228227, P = 257;
ld const EPS = 1e-10;

void FastIO() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
}

struct Node {
    ll x, sz, pr;
    Node* left = nullptr;
    Node* right = nullptr;
    Node() {}
    Node(ll _x) {
        ll x = _x;
        sz = 1;
        pr = rand();
    }
};

ll GetSz(Node* tree) {
    return (tree ? tree->sz : 0);
}

void Upd(Node* tree) {
    tree->sz = GetSz(tree->left) + GetSz(tree->right) + 1;
}

Node* Merge(Node* left, Node* right) {
    if (! left) {
        return right;
    }
    if (! right) {
        return left;
    }

    if (left->pr > right->pr) {
        left->right = Merge(left->right, right);
        Upd(left);
        return left;
    }
    else {
        right->left = Merge(left, right->left);
        Upd(right);
        return right;
    }
}

pair<Node*, Node*> Split(Node* tree, ll x) {
    if (! tree) {
        return {tree, tree};
    }

    if (GetSz(tree->left) == x - 1) {
        auto tmp = tree->right;
        tree->right = nullptr;
        Upd(tree);
        return {tree, tmp};
    }
    else if (GetSz(tree->left) > x - 1) {
```

```
        auto tmp = Split(tree->left, x);
        tree->left = tmp.second;
        Upd(tree);
        return {tmp.first, tree};
    }
    else {
        auto tmp = Split(tree->right, x - GetSz(tree->left) - 1);
        tree->right = tmp.first;
        Upd(tree);
        return {tree, tmp.second};
    }
}

Node* Ins(Node* tree, ll x, ll pos) {
    Node* node = new Node(x);
    auto tmp = Split(tree, pos - 1);
    return Merge(tmp.first, Merge(node, tmp.second));
}

void Zfunc(string s) {
    ll n = s.size();
    vector<ll> z(n, 0);
    ll l = 0, r = 0;
    for (int i = 1; i < n; ++i) {
        if (r >= i) {
            z[i] = min(r - i + 1, z[i - l]);
        }
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
            ++z[i];
        }
        if (i + z[i] - 1 > r) {
            r = i + z[i] - 1;
            l = i;
        }
    }
}

ll bn(ll x, ll n, ll m) {
    if (n == 0) {
        return 1;
    }

    if (n % 2) {
        return (bn(x, n - 1, m) * x) % m;
    }
    else {
        ll tmp = bn(x, n / 2, m);
        return (tmp * tmp) % m;
    }
}

ll gcd(ll a, ll b) {
    while (a && b) {
        if (a > b) {
            a = a % b;
        }
        else {
            b = b % a;
        }
    }
    return a + b;
}

signed main() {
    FastIO();

    srand(time(NULL));

    string s;
    cin >> s;
    ll n;
    cin >> n;

    set<char> st;
```

```
    for (int i = 0; i < s.size(); ++i) {
        st.insert(s[i]);
    }

    ll ans = 0;
    for (int i = 0; i < n; ++i) {
        string t;
        cin >> t;
        ll ind = 0, mx = 0;
        while (ind < t.size()) {
            if (st.find(t[ind]) == st.end()) {
                ++ind;
            }
            else {
                ll cur1 = 0, cur2 = ind, cnt = 0;
                while (cur1 < s.size() && cur2 < t.size()) {
                    if (s[cur1] == t[cur2]) {
                        ++cnt;
                        ++cur2;
                    }
                    ++cur1;
                }
                mx = max(mx, cnt);
                ++ind;
            }
        }

        ans += s.size() - mx;
    }

    cout << ans;

    return 0;
}
```

Task D ()

```
//#pragma GCC optimize("Ofast,no-stack-protector,unroll-loops,fast-math")
//#pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,avx,avx2,tune=native")

#include <bits/stdc++.h>

using namespace std;

#define ll long long
#define ld long double
#define pll pair<ll, ll>
#define point pair<ld, ld>

ll const N = 1e3 + 1, M = 1e9 + 7, P = 257, INF = 1e18;
ld const EPS = 1e-10;

void FastIO() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
}

struct Node {
    ll x, sz, pr;
    Node* left = nullptr;
    Node* right = nullptr;
    Node() {}
    Node(ll _x) {
        ll x = _x;
        sz = 1;
        pr = rand();
    }
};

ll GetSz(Node* tree) {
    return (tree ? tree->sz : 0);
}

void Upd(Node* tree) {
    tree->sz = GetSz(tree->left) + GetSz(tree->right) + 1;
}

Node* Merge(Node* left, Node* right) {
    if (! left) {
        return right;
    }
    if (! right) {
        return left;
    }

    if (left->pr > right->pr) {
        left->right = Merge(left->right, right);
        Upd(left);
        return left;
    }
    else {
        right->left = Merge(left, right->left);
        Upd(right);
        return right;
    }
}

pair<Node*, Node*> Split(Node* tree, ll x) {
    if (! tree) {
        return {tree, tree};
    }

    if (GetSz(tree->left) == x - 1) {
        auto tmp = tree->right;
        tree->right = nullptr;
        Upd(tree);
        return {tree, tmp};
    }
```

```
    else if (GetSz(tree->left) > x - 1) {
        auto tmp = Split(tree->left, x);
        tree->left = tmp.second;
        Upd(tree);
        return {tmp.first, tree};
    }
    else {
        auto tmp = Split(tree->right, x - GetSz(tree->left) - 1);
        tree->right = tmp.first;
        Upd(tree);
        return {tree, tmp.second};
    }
}

Node* Ins(Node* tree, ll x, ll pos) {
    Node* node = new Node(x);
    auto tmp = Split(tree, pos - 1);
    return Merge(tmp.first, Merge(node, tmp.second));
}

void Zfunc(string s) {
    ll n = s.size();
    vector<ll> z(n, 0);
    ll l = 0, r = 0;
    for (int i = 1; i < n; ++i) {
        if (r >= i) {
            z[i] = min(r - i + 1, z[i - l]);
        }
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
            ++z[i];
        }
        if (i + z[i] - 1 > r) {
            r = i + z[i] - 1;
            l = i;
        }
    }
}

ll bn(ll x, ll n, ll m) {
    if (n == 0) {
        return 1;
    }

    if (n % 2) {
        return (bn(x, n - 1, m) * x) % m;
    }
    else {
        ll tmp = bn(x, n / 2, m);
        return (tmp * tmp) % m;
    }
}

ll gcd(ll a, ll b) {
    while (a && b) {
        if (a > b) {
            a = a % b;
        }
        else {
            b = b % a;
        }
    }
    return a + b;
}

ld kos(point a, point b, point c) {
    return ((a.first - c.first) * (b.second - c.second) - (b.first - c.first) * (a.second - c.
        second));
}

pll t[N][N];
ll dp[N][N];
bool used[N][N];

signed main() {
```

```
FastIO();

srand(time(NULL));

//cout << setprecision(10) << fixed;

//freopen("input.txt", "r", stdin);

ll n, m;
cin >> n >> m;
ll ar, br, ac, bc;
cin >> ar >> br >> ac >> bc;

for (int i = 0; i < n; ++i) {
    for (int j = 0; j < m; ++j) {
        ll x, y;
        cin >> x >> y;
        t[i][j] = {x, y};
        dp[i][j] = INF;
    }
}

--br;
--bc;
--ar;
--ac;
dp[ar][br] = 0;

for (int k = 0; k < m * n; ++k) {
    ll minx = -1, miny = -1;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            if (! used[i][j]) {
                if (minx == -1) {
                    miny = i;
                    minx = j;
                }
                else {
                    if (dp[miny][minx] > dp[i][j]) {
                        miny = i;
                        minx = j;
                    }
                }
            }
        }
    }

    used[miny][minx] = true;

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            if (used[i][j]) {
                continue;
            }

            ll extra = 0;
            if (minx == j) {
                extra += abs(t[miny][minx].second);
            }
            else {
                extra += (abs(minx - j) / (minx - j)) * t[miny][minx].second;
            }

            if (miny == i) {
                extra += abs(t[miny][minx].first);
            }
            else {
                extra += (abs(miny - i) / (miny - i)) * t[miny][minx].first;
            }

            dp[i][j] = min(dp[i][j], dp[miny][minx] + abs(miny - i) + abs(minx - j) + extra);
        }
    }
}
```

```
    cout << dp[ac][bc];

    return 0;
}
```

Task E ()

```
//#pragma GCC optimize("Ofast,no-stack-protector,unroll-loops,fast-math")
//#pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,avx,avx2,tune=native")

#include <bits/stdc++.h>

using namespace std;

#define ll long long
#define ld long double
#define pll pair<ll, ll>
#define point pair<ld, ld>

ll const N = 1e3 + 1, M = 1e9 + 7, P = 257, INF = 1e18;
ld const EPS = 1e-10;

void FastIO() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
}

struct Node {
    ll x, sz, pr;
    Node* left = nullptr;
    Node* right = nullptr;
    Node() {}
    Node(ll _x) {
        ll x = _x;
        sz = 1;
        pr = rand();
    }
};

ll GetSz(Node* tree) {
    return (tree ? tree->sz : 0);
}

void Upd(Node* tree) {
    tree->sz = GetSz(tree->left) + GetSz(tree->right) + 1;
}

Node* Merge(Node* left, Node* right) {
    if (! left) {
        return right;
    }
    if (! right) {
        return left;
    }

    if (left->pr > right->pr) {
        left->right = Merge(left->right, right);
        Upd(left);
        return left;
    }
    else {
        right->left = Merge(left, right->left);
        Upd(right);
        return right;
    }
}

pair<Node*, Node*> Split(Node* tree, ll x) {
    if (! tree) {
        return {tree, tree};
    }

    if (GetSz(tree->left) == x - 1) {
        auto tmp = tree->right;
        tree->right = nullptr;
        Upd(tree);
        return {tree, tmp};
```

```
    }
    else if (GetSz(tree->left) > x - 1) {
        auto tmp = Split(tree->left, x);
        tree->left = tmp.second;
        Upd(tree);
        return {tmp.first, tree};
    }
    else {
        auto tmp = Split(tree->right, x - GetSz(tree->left) - 1);
        tree->right = tmp.first;
        Upd(tree);
        return {tree, tmp.second};
    }
}

Node* Ins(Node* tree, ll x, ll pos) {
    Node* node = new Node(x);
    auto tmp = Split(tree, pos - 1);
    return Merge(tmp.first, Merge(node, tmp.second));
}

void Zfunc(string s) {
    ll n = s.size();
    vector<ll> z(n, 0);
    ll l = 0, r = 0;
    for (int i = 1; i < n; ++i) {
        if (r >= i) {
            z[i] = min(r - i + 1, z[i - l]);
        }
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
            ++z[i];
        }
        if (i + z[i] - 1 > r) {
            r = i + z[i] - 1;
            l = i;
        }
    }
}

ll bn(ll x, ll n, ll m) {
    if (n == 0) {
        return 1;
    }

    if (n % 2) {
        return (bn(x, n - 1, m) * x) % m;
    }
    else {
        ll tmp = bn(x, n / 2, m);
        return (tmp * tmp) % m;
    }
}

ll gcd(ll a, ll b) {
    while (a && b) {
        if (a > b) {
            a = a % b;
        }
        else {
            b = b % a;
        }
    }
    return a + b;
}

signed main() {
    //FastIO();

    srand(time(NULL));

    //cout << setprecision(10) << fixed;

    //freopen("input.txt", "r", stdin);
```

```
    ll n, m, b;
    cin >> n >> m >> b;

    bool t[n][m];
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            t[i][j] = false;
        }
    }

    /*for (int i = 0; i < b; ++i) {
        ll x, y;
        cin >> x >> y;
        --x;
        --y;
        t[x][y] = true;
    }*/

    if (b == 1) {
        ll x, y;
        cin >> y >> x;

        set<pll> s;
        cout << "? 0 0 100 100" << endl;
        s.insert({0, 0});
        s.insert({100, 100});
        ll a, c;
        cin >> a >> c;
        s.erase({a, c});
        auto tmp = *s.begin();
        cout << "! " << tmp.second - y + 1 << ' ' << tmp.first - x + 1;
    }
    else {
        ll x, y, w, z;
        cin >> y >> x >> z >> w;
        set<pll> s;
        s.insert({0, 0});
        s.insert({100, 100});
        cout << "? 0 0 100 100" << endl;
        s.insert({200, 200});
        s.insert({300, 300});
        ll a, c;
        cin >> a >> c;
        s.erase({a, c});
        cout << "? 200 200 300 300" << endl;
        cin >> a >> c;
        s.erase({a, c});
        auto o = *s.begin();
        auto k = *s.rbegin();
        cout << "? " << o.first + y - z << ' ' << o.second + x - w << ' ' << k.first + y - z << '
            ' << k.second + x - w << endl;
        cin >> a >> c;
        if (a == o.first + y - z || a == o.first) {
            cout << "! " << k.first - z + 1 << ' ' << k.second - w + 1;
        }
        else {
            cout << "! " << o.first - z + 1 << ' ' << o.second - w + 1;
        }
    }

    return 0;
}
```

Task F ()

```
//#pragma GCC optimize("Ofast,no-stack-protector,unroll-loops,fast-math")
//#pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,avx,avx2,tune=native")

#include <bits/stdc++.h>

using namespace std;

#define ll long long
#define ld long double
#define pll pair<ll, ll>
#define point pair<ld, ld>

ll const N = 1e3 + 1, M = 1e9 + 7, P = 257, INF = 1e18;
ld const EPS = 1e-10;

void FastIO() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
}

struct Node {
    ll x, sz, pr;
    Node* left = nullptr;
    Node* right = nullptr;
    Node() {}
    Node(ll _x) {
        ll x = _x;
        sz = 1;
        pr = rand();
    }
};

ll GetSz(Node* tree) {
    return (tree ? tree->sz : 0);
}

void Upd(Node* tree) {
    tree->sz = GetSz(tree->left) + GetSz(tree->right) + 1;
}

Node* Merge(Node* left, Node* right) {
    if (! left) {
        return right;
    }
    if (! right) {
        return left;
    }

    if (left->pr > right->pr) {
        left->right = Merge(left->right, right);
        Upd(left);
        return left;
    }
    else {
        right->left = Merge(left, right->left);
        Upd(right);
        return right;
    }
}

pair<Node*, Node*> Split(Node* tree, ll x) {
    if (! tree) {
        return {tree, tree};
    }

    if (GetSz(tree->left) == x - 1) {
        auto tmp = tree->right;
        tree->right = nullptr;
        Upd(tree);
        return {tree, tmp};
    }
```

```
    else if (GetSz(tree->left) > x - 1) {
        auto tmp = Split(tree->left, x);
        tree->left = tmp.second;
        Upd(tree);
        return {tmp.first, tree};
    }
    else {
        auto tmp = Split(tree->right, x - GetSz(tree->left) - 1);
        tree->right = tmp.first;
        Upd(tree);
        return {tree, tmp.second};
    }
}

Node* Ins(Node* tree, ll x, ll pos) {
    Node* node = new Node(x);
    auto tmp = Split(tree, pos - 1);
    return Merge(tmp.first, Merge(node, tmp.second));
}

void Zfunc(string s) {
    ll n = s.size();
    vector<ll> z(n, 0);
    ll l = 0, r = 0;
    for (int i = 1; i < n; ++i) {
        if (r >= i) {
            z[i] = min(r - i + 1, z[i - l]);
        }
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
            ++z[i];
        }
        if (i + z[i] - 1 > r) {
            r = i + z[i] - 1;
            l = i;
        }
    }
}

ll bn(ll x, ll n, ll m) {
    if (n == 0) {
        return 1;
    }

    if (n % 2) {
        return (bn(x, n - 1, m) * x) % m;
    }
    else {
        ll tmp = bn(x, n / 2, m);
        return (tmp * tmp) % m;
    }
}

ll gcd(ll a, ll b) {
    while (a && b) {
        if (a > b) {
            a = a % b;
        }
        else {
            b = b % a;
        }
    }
    return a + b;
}

signed main() {
    //FastIO();

    srand(time(NULL));

    //cout << setprecision(10) << fixed;

    //freopen("input.txt", "r", stdin);

    ll n, m;
```

```
    cin >> n >> m;
    if (n == 2) {
        cout << "1";
    }
    else if (n == 3) {
        cout << "0 0 0 3";
    }
    else if (n == 4) {
        cout << "0 0 0 0 0 0 0 0 4 12";
    }
    else if (n == 5) {
        cout << "0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 60 0 60";
    }
    else if (n == 6) {
        cout << "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 120 150 0 0 360 0 0 360";
    }

    return 0;
}
```