Олимпиада СПбГУ по информатике 2022/23 учебного года

A	B	C	D	E	F	Sum
100	100	100	10	28	65	403

Task A ()

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

using namespace std;
using ll = long long;

void resolve() {
    ll n = 6;
    vector<ll> order;
    for (ll i = 0; i < n; ++i) {
        ll pos;
        cin >> pos;
        --pos;
        order.insert(order.begin() + pos, i);
    }

    vector<ll> res(n);
    for (ll i = 0; i < n; ++i) {
        res[order[i]] = i;
    }

    for (ll i = 0; i < n; ++i) {
        cout << res[i] + 1 << ' ';
    }
    cout << '\n';
}

int32_t main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    resolve();
    return 0;
}
```

Task B ()

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

using namespace std;
using ll = long long;

void solveFirst() {
    ll n;
    cin >> n;
    ll sum = 0;
    for (ll i = 0; i < n; ++i) {
        ll x;
        cin >> x;
        sum += x;
    }

    cout << sum * 1000 << '\n';
}

void solveSecond() {
    ll n;
    cin >> n;
    ll x;
    cin >> x;
    ll prevSum = x / 1000;
    ll res = prevSum + x - prevSum * 1000;
    for (ll i = 1; i < n; ++i) {
        cin >> x;
        res += x - prevSum * 1000;
    }

    cout << res << '\n';
}

void resolve() {
    string type;
    cin >> type;
    if (type[0] == 'f') {
        solveFirst();
    } else {
        solveSecond();
    }
}

int32_t main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    resolve();
    return 0;
}
```

Task C ()

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

using namespace std;
using ll = long long;

void resolve() {
    ll n = 3;
    vector<pair<ll, ll>> src(n);
    for (auto& item : src) {
        cin >> item.first >> item.second;
        if (item.first > item.second) {
            swap(item.first, item.second);
        }
    }

    vector<ll> order = {0, 1, 2};
    set<pair<ll, ll>> res;
    do {
        pair<ll, ll> init = src[order[0]];
        ll N = init.first;
        ll M = init.second;
        res.insert({min(N, M), max(N, M)});

        pair<ll, ll> sec = src[order[1]];
        for (ll j = 0; j < 2; ++j) {
            if (sec.first > N || sec.second > M) {
                continue;
            }

            if (sec.first == N) {
                if (sec.second < M) {
                    res.insert({min(N, M - sec.second), max(N, M - sec.second)});
                }
            }

            if (sec.second == M) {
                if (sec.first < N) {
                    res.insert({min(N - sec.first, M), max(N - sec.first, M)});
                }
            }

            vector<pair<ll, ll>> angles = {
                    {0,              0},
                    {0,              M - sec.second},
                    {N - sec.first, 0},
                    {N - sec.first, M - sec.second}
            };

            for (const auto& item : angles) {
                vector<vector<ll>> field(N, vector<ll>(M));
                for (ll i = 0; i < sec.first; ++i) {
                    for (ll k = 0; k < sec.second; ++k) {
                        field[item.first + i][item.second + k] = 1;
                    }
                }

                pair<ll, ll> thi = src[order[2]];
                for (ll i = 0; i < 2; ++i) {
                    if (thi.first > N || thi.second > M) {
                        continue;
                    }

                    for (ll tI = 0; tI < N; ++tI) {
                        for (ll tJ = 0; tJ < M; ++tJ) {
                            if (tI + thi.first > N || tJ + thi.second > M) {
                                break;
                            }

                            vector<vector<ll>> fiT = field;
```

```
bool needBreak = false;
for (int k = 0; k < thi.first; ++k) {
    for (int l = 0; l < thi.second; ++l) {
        if (field[tI + k][tJ + l] == 1) {
            needBreak = true;
            break;
        }
        fiT[tI + k][tJ + l] = 2;
    }
}

if (needBreak) {
    continue;
}

bool canGetHor = (tI == 0) || (tI == N - thi.first);
bool canGetVer = tJ == 0 || tJ == M - thi.second;
if (!canGetHor) {
    for (ll k = 0; k < thi.second; ++k) {
        if (fiT[tI - 1][tJ + k] == 1) {
            canGetHor = true;
            break;
        }
    }

    if (!canGetHor) {
        for (ll k = 0; k < thi.second; ++k) {
            if (fiT[tI + thi.first][tJ + k] == 1) {
                canGetHor = true;
                break;
            }
        }
    }
}

if (!canGetVer) {
    for (ll k = 0; k < thi.first; ++k) {
        if (fiT[tI + k][tJ - 1] == 1) {
            canGetVer = true;
            break;
        }
    }

    if (!canGetVer) {
        for (ll k = 0; k < thi.first; ++k) {
            if (fiT[tI + k][tJ + thi.second] == 1) {
                canGetVer = true;
                break;
            }
        }
    }
}

if (!canGetHor || !canGetVer) {
    continue;
}

vector<vector<vector<ll>>> pref(N, vector<vector<ll>>(M, vector<ll>(3)
    ));
pref[0][0][fiT[0][0]] = 1;
for (ll k = 1; k < N; ++k) {
    pref[k][0] = pref[k - 1][0];
    ++pref[k][0][fiT[k][0]];
}
for (ll k = 1; k < M; ++k) {
    pref[0][k] = pref[0][k - 1];
    ++pref[0][k][fiT[0][k]];
}

for (ll k = 1; k < N; ++k) {
    for (ll l = 1; l < M; ++l) {
        for (ll m = 0; m < 3; ++m) {
            pref[k][l][m] = pref[k - 1][l][m] + pref[k][l - 1][m] -
                pref[k - 1][l - 1][m];
```

```
                ++pref[k][l][fiT[k][l]];
            }
        }
    }

    for (ll sI = 0; sI < N; ++sI) {
        for (ll sJ = 0; sJ < M; ++sJ) {
            for (ll fI = sI; fI < N; ++fI) {
                for (ll fJ = sJ; fJ < M; ++fJ) {
                    vector<ll> pr = pref[fI][fJ];
                    if (sI) {
                        for (ll k = 0; k < 3; ++k) {
                            pr[k] -= pref[sI - 1][fJ][k];
                        }
                    }
                    if (sJ) {
                        for (ll k = 0; k < 3; ++k) {
                            pr[k] -= pref[fI][sJ - 1][k];
                        }
                    }
                    if (sI && sJ) {
                        for (ll k = 0; k < 3; ++k) {
                            pr[k] += pref[sI - 1][sJ - 1][k];
                        }
                    }

                    if (!pr[0] + !pr[1] + !pr[2] != 2) {
                        continue;
                    }

                    bool isFit = true;
                    if (sI) {
                        for (ll k = sJ; k <= fJ; ++k) {
                            if (fiT[sI - 1][k] == fiT[sI][sJ]) {
                                isFit = false;
                                break;
                            }
                        }
                    }
                    if (sJ) {
                        for (ll k = sI; k <= fI; ++k) {
                            if (fiT[k][sJ - 1] == fiT[sI][sJ]) {
                                isFit = false;
                                break;
                            }
                        }
                    }
                    if (fI < N - 1) {
                        for (ll k = sJ; k <= fJ; ++k) {
                            if (fiT[fI + 1][k] == fiT[sI][sJ]) {
                                isFit = false;
                                break;
                            }
                        }
                    }
                    if (fJ < M - 1) {
                        for (ll k = sI; k <= fI; ++k) {
                            if (fiT[k][fJ + 1] == fiT[sI][sJ]) {
                                isFit = false;
                                break;
                            }
                        }
                    }

                    if (isFit) {
                        res.insert({
                                        min(fI - sI + 1, fJ - sJ + 1),
                                        max(fI - sI + 1, fJ - sJ + 1)
                                });
                    }
                }
            }
        }
    }
```

```
                        }
                    }

                    swap(thi.first, thi.second);
                }
            }

            swap(sec.first, sec.second);
        }
    } while (next_permutation(order.begin(), order.end()));

    for (const auto& item : res) {
        cout << item.first << ' ' << item.second << '\n';
    }
}

int32_t main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    resolve();
    return 0;
}
```

Task D ()

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

using namespace std;
using ll = long long;

const ll maxA = 60;

void solveN1() {
    ll x;
    cin >> x;
    cout << 1 << ' ' << x << endl;
    ll id, val;
    cin >> id >> val;
    if (id == -1) {
        exit(0);
    }

    cout << 1 << ' ' << x << endl;
    exit(0);
}

void generate(map<vector<ll>, vector<ll>>& states, vector<ll>& state, const vector<ll>& init) {
    states[state] = {-1, -1, -1, -1};
    if (!state[0] && !state[1] && !state[2] && !state[3]) {
        return;
    }

    vector<ll> next;
    if (state[0]) {
        next = {0, state[1], init[0], state[3]};
        if (states.find(next) == states.end()) {
            generate(states, next, init);
        }
        if (states[next][0] == -1) {
            states[state] = next;
        }
    }
    if (state[1]) {
        next = {state[0], 0, state[2], init[1]};
        if (states.find(next) == states.end()) {
            generate(states, next, init);
        }
        if (states[next][0] == -1) {
            states[state] = next;
        }
    }
    if (state[2]) {
        next = state;
        for (ll i = 1; i <= state[2]; ++i) {
            next[2] = state[2] - i;
            if (states.find(next) == states.end()) {
                generate(states, next, init);
            }
            if (states[next][0] == -1) {
                states[state] = next;
            }
        }
    }
    if (state[3]) {
        next = state;
        for (ll i = 1; i <= state[3]; ++i) {
            next[3] = state[3] - i;
            if (states.find(next) == states.end()) {
                generate(states, next, init);
            }
            if (states[next][0] == -1) {
                states[state] = next;
            }
        }
    }
```

```
}

void solveTree() {
    ll n = 2;
    vector<ll> src(n);
    for (auto& item : src) {
        cin >> item;
    }

    vector<ll> init = src;

    map<vector<ll>, vector<ll>> steps;
    vector<ll> initState = {1, 1, src[0], src[1]};
    generate(steps, initState, init);
    if (steps[{1, 1, src[0], src[1]}][0] == -1) {
        cout << -1 << ' ' << -1 << endl;
        exit(0);
    }

    vector<bool> can(n, true);
    while (true) {
        vector<ll> step = steps[{can[0], can[1], src[0], src[1]}];
        if (can[0] && !step[0]) {
            cout << 1 << ' ' << 0 << endl;
            can[0] = false;
            src[0] = step[0];
        } else if (can[1] && !step[1]) {
            cout << 2 << ' ' << 0 << endl;
            can[1] = false;
            src[1] = step[1];
        } else if (src[0] != step[2]) {
            cout << 1 << ' ' << src[0] - step[2] << endl;
            src[0] = step[2];
        } else {
            cout << 2 << ' ' << src[1] - step[3] << endl;
            src[1] = step[3];
        }

        ll id, act;
        cin >> id >> act;
        if (id == -1) {
            exit(0);
        }

        --id;
        if (act == 0) {
            src[id] = init[id];
            can[id] = false;
        } else {
            src[id] -= act;
        }
    }
}

void resolve() {
    vector<ll> dpUsed(maxA + 1);
    vector<vector<ll>> dpNotUsed(maxA + 1, vector<ll>(maxA + 1));
    for (ll sz = 1; sz <= maxA; ++sz) {
        vector<bool> mex(maxA + 1);
        for (ll take = 1; take <= sz; ++take) {
            mex[dpUsed[sz - take]] = true;
        }

        for (ll j = 0; j <= maxA; ++j) {
            if (!mex[j]) {
                dpUsed[sz] = j;
                break;
            }
        }
    }

    for (ll initSz = 0; initSz <= maxA; ++initSz) {
        for (ll sz = 0; sz <= initSz; ++sz) {
            vector<bool> mex(maxA + 1);
```

```
            for (ll take = 1; take <= sz; ++take) {
                mex[dpNotUsed[initSz][sz - take]] = true;
            }

            mex[dpUsed[initSz]] = true;

            for (ll j = 0; j <= maxA; ++j) {
                if (!mex[j]) {
                    dpNotUsed[initSz][sz] = j;
                    break;
                }
            }
        }
    }

    ll n;
    cin >> n;
    if (n == 1) {
        solveN1();
        exit(0);
    }

//     if (n == 2) {
//         solveTree();
//         exit(0);
//     }

    vector<ll> initSZs(n);
    vector<ll> sZs(n);
    vector<bool> used(n);
    ll xorSum = 0;
    for (ll i = 0; i < n; ++i) {
        cin >> initSZs[i];
        sZs[i] = initSZs[i];
        xorSum ^= dpNotUsed[initSZs[i]][initSZs[i]];
    }

    if (xorSum == 0) {
        cout << "-1 -1" << endl;
        exit(0);
    }

    while (true) {
        xorSum = 0;
        for (ll i = 0; i < n; ++i) {
            if (used[i]) {
                xorSum ^= dpUsed[sZs[i]];
            } else {
                xorSum ^= dpNotUsed[initSZs[i]][sZs[i]];
            }
        }

        for (ll id = 0; id < n; ++id) {
            if (!used[id]) {
                if (((xorSum ^ dpNotUsed[initSZs[id]][sZs[id]]) ^ dpUsed[initSZs[id]]) == 0) {
                    cout << id + 1 << ' ' << 0 << endl;
                    sZs[id] = initSZs[id];
                    used[id] = true;
                    break;
                }
            }

            bool printed = false;
            for (ll take = 1; take <= sZs[id]; ++take) {
                ll newSum;
                if (used[id]) {
                    newSum = ((xorSum ^ dpUsed[sZs[id]]) ^ dpUsed[sZs[id] - take]);
                } else {
                    newSum = ((xorSum ^ dpNotUsed[initSZs[id]][sZs[id]]) ^ dpNotUsed[initSZs[id]][
                        sZs[id] - take]);
                }

                if (newSum == 0) {
                    cout << id + 1 << ' ' << take << endl;
```

```
                    sZs[id] -= take;
                    printed = true;
                    break;
                }
            }

            if (printed) {
                break;
            }
        }

        ll id, act;
        cin >> id >> act;
        if (id == -1) {
            exit(0);
        }

        --id;
        if (act == 0) {
            sZs[id] = initSZs[id];
            used[id] = true;
        } else {
            sZs[id] -= act;
        }
    }
}

int32_t main() {
    ios_base::sync_with_stdio(false);
    resolve();
    return 0;
}
```

Task E ()

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <unordered_map>
#include <set>

using namespace std;
using ll = long long;

unordered_map<ll, ll> refer;
unordered_map<ll, ll> antiRefer;

void transmit() {
    ll n;
    cin >> n;
    ll need = antiRefer[n];
    vector<ll> digs;
    while (need) {
        digs.push_back(need % 10);
        need /= 10;
    }

    for (ll i = 0; i < 10; ++i) {
        for (ll j = 0; j < digs[i]; ++j) {
            cout << '1';
        }
        for (ll j = digs[i]; j < 10; ++j) {
            cout << '0';
        }
        cout << '\n';
    }
}

void receive() {
    vector<ll> digs;
    for (ll i = 0; i < 10; ++i) {
        string src;
        cin >> src;
        digs.push_back(count(src.begin(), src.end(), '1'));
    }

    sort(digs.begin(), digs.end());
    ll p = 1;
    ll from = 0;
    for (ll i = 0; i < 10; ++i) {
        from += digs[i] * p;
        p *= 10;
    }

    cout << refer[from] << '\n';
}

void resolve() {
    ll curX = 1;
    ll base = 1e9;
    set<ll> was;
    for (ll i = base; i < base + 6000000; ++i) {
        vector<ll> digs;
        ll x = i;
        while (x) {
            digs.push_back(x % 10);
            x /= 10;
        }

        sort(digs.begin(), digs.end());
        ll p = 1;
        ll from = 0;
        for (ll ii = 0; ii < 10; ++ii) {
            from += digs[ii] * p;
            p *= 10;
        }
```

```
        if (was.find(from) != was.end()) {
            continue;
        }

        was.insert(from);
        refer[from] = curX;
        antiRefer[curX] = from;
        ++curX;
    }

    ll testN;
    cin >> testN;
    string type;
    cin >> type;
    while (testN--) {
        if (type[0] == 't') {
            transmit();
        } else {
            receive();
        }
    }
}

int32_t main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    resolve();
    return 0;
}
```

Task F ()