Олимпиада СПбГУ по информатике 2020/21 учебного года

A	B	C	D	E	F	Sum
100	100	100	100	55	25	480

Task A ()

```
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <vector>
#include <set>
#include <bitset>
#include <map>
#include <string>
#include <ctime>
#include <queue>
#include <numeric>
#include <random>
#include <immintrin.h>
/*
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.2,avx,popcnt,abm,mmx,tune=native")
#pragma GCC target("avx2")
#pragma GCC optimize("Ofast,fast-math,no-stack-protector,unroll-loops,inline")
*/
/*
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.1,sse4.2,popcnt,abm,mmx,avx,avx2,tune=native")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("profile-values,profile-reorder-functions,tracer")
#pragma GCC optimize("vpt")
#pragma GCC optimize("rename-registers")
#pragma GCC optimize("move-loop-invariants")
#pragma GCC optimize("unswitch-loops")
#pragma GCC optimize("function-sections")
#pragma GCC optimize("data-sections")
#pragma GCC optimize("branch-target-load-optimize")
#pragma GCC optimize("branch-target-load-optimize2")
#pragma GCC optimize("btr-bb-exclusive")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-falign-functions")
```

```
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
*/
// #define int long long
using namespace std;
typedef long double ld;
typedef long long ll;
const int inf = 1e9 + 7;
const ll l_inf = (ll) inf * (ll) inf;
const ld eps = 1e-6;


ll f(int k, int depth = 0) {
    if (k == 0 || depth == 100) {
        return 0;
    }
    return k + f(k - 1, depth + 1) / 10;
}


signed main() {
#ifdef LOCAL
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
#else
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#endif
    int k;
    cin >> k;
    cout << f(k) % 10 << "\n";
    cerr << 1.0 * clock() / CLOCKS_PER_SEC << "\n";
    return 0;
}
```

Task B ()

```
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <vector>
#include <set>
#include <bitset>
#include <map>
#include <string>
#include <ctime>
#include <queue>
#include <numeric>
#include <random>
#include <immintrin.h>
/*
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.2,avx,popcnt,abm,mmx,tune=native")
#pragma GCC target("avx2")
#pragma GCC optimize("Ofast,fast-math,no-stack-protector,unroll-loops,inline")
*/
/*
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.1,sse4.2,popcnt,abm,mmx,avx,avx2,tune=native")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("profile-values,profile-reorder-functions,tracer")
#pragma GCC optimize("vpt")
#pragma GCC optimize("rename-registers")
#pragma GCC optimize("move-loop-invariants")
#pragma GCC optimize("unswitch-loops")
#pragma GCC optimize("function-sections")
#pragma GCC optimize("data-sections")
#pragma GCC optimize("branch-target-load-optimize")
#pragma GCC optimize("branch-target-load-optimize2")
#pragma GCC optimize("btr-bb-exclusive")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
*/
```

```
// #define int long long
using namespace std;
typedef long double ld;
typedef long long ll;
const int inf = 1e9 + 7;
const ll l_inf = (ll) inf * (ll) inf;
const ld eps = 1e-6;
const int A = 26;


signed main() {
#ifdef LOCAL
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
#else
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#endif
    int n, k;
    cin >> n >> k;
    string S;
    cin >> S;
    vector<int> dp(n + 1, inf);
    dp[0] = 0;
    int j = 1;
    vector<int> cnt(A, 0);
    int cnt_diff = 0;
    for (int i = 1; i <= n; ++i) {
        cnt_diff += cnt[S[i - 1] - 'a'] == 0;
        ++cnt[S[i - 1] - 'a'];
        while (i - j + 1 > k || cnt_diff > 3) {
            cnt_diff -= cnt[S[j - 1] - 'a'] == 1;
            --cnt[S[j - 1] - 'a'];
            ++j;
        }
        dp[i] = dp[j - 1] + 1;
    }
    cout << dp[n] << "\n";
    cerr << 1.0 * clock() / CLOCKS_PER_SEC << "\n";
    return 0;
}
```

Task C ()

```
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <vector>
#include <set>
#include <bitset>
#include <map>
#include <string>
#include <ctime>
#include <queue>
#include <numeric>
#include <random>
#include <immintrin.h>
/*
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.2,avx,popcnt,abm,mmx,tune=native")
#pragma GCC target("avx2")
#pragma GCC optimize("Ofast,fast-math,no-stack-protector,unroll-loops,inline")
*/
/*
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.1,sse4.2,popcnt,abm,mmx,avx,avx2,tune=native")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("profile-values,profile-reorder-functions,tracer")
#pragma GCC optimize("vpt")
#pragma GCC optimize("rename-registers")
#pragma GCC optimize("move-loop-invariants")
#pragma GCC optimize("unswitch-loops")
#pragma GCC optimize("function-sections")
#pragma GCC optimize("data-sections")
#pragma GCC optimize("branch-target-load-optimize")
#pragma GCC optimize("branch-target-load-optimize2")
#pragma GCC optimize("btr-bb-exclusive")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
*/
```

```cpp
// #define int long long
using namespace std;
typedef long double ld;
typedef long long ll;
const int inf = 1e9 + 7;
const ll l_inf = (ll) inf * (ll) inf;
const ld eps = 1e-6;
const int MAXN = 500, MAXX = 25e4;

bitset<MAXX + 1> P[MAXN + 1];


signed main() {
#ifdef LOCAL
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
#else
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#endif
    int n, x, y;
    cin >> n >> x >> y;
    vector<int> V(n), W(n);
    for (int i = 0; i < n; ++i) {
        cin >> V[i];
    }
    for (int i = 0; i < n; ++i) {
        cin >> W[i];
    }
    //
    vector<int> dp(x + 1, inf);
    dp[0] = 0;
    for (int i = 1; i <= n; ++i) {
        vector<int> dp_(x + 1, inf);
        for (int v = 0; v <= x; ++v) {
            dp_[v] = dp[v] + W[i - 1];
            if (v - V[i - 1] >= 0 && dp[v - V[i - 1]] < dp_[v]) {
                dp_[v] = dp[v - V[i - 1]];
                P[i][v] = true;
            }
        }
        dp = dp_;
    }
    //
    pair<int, int> best = make_pair(inf, inf);
    for (int v = 0; v <= x; ++v) {
        best = min(best, make_pair(dp[v], v));
    }
    //
    if (best.first > y) {
        cout << "-1\n";
        return 0;
    }
    int v = best.second;
    vector<bool> res(n);
    for (int i = n; i >= 1; --i) {
        res[i - 1] = P[i][v];
        if (P[i][v]) {
            v -= V[i - 1];
        }
    }
    //
    for (int i = 0; i < n; ++i) {
        if (res[i]) {
            cout << "x";
        } else {
            cout << "y";
        }
    }
    cout << "\n";
    cerr << 1.0 * clock() / CLOCKS_PER_SEC << "\n";
    return 0;
}
```

Task D ()

```
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <vector>
#include <set>
#include <bitset>
#include <map>
#include <string>
#include <ctime>
#include <queue>
#include <numeric>
#include <random>
#include <immintrin.h>
/*
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.2,avx,popcnt,abm,mmx,tune=native")
#pragma GCC target("avx2")
#pragma GCC optimize("Ofast,fast-math,no-stack-protector,unroll-loops,inline")
*/
/*
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.1,sse4.2,popcnt,abm,mmx,avx,avx2,tune=native")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("profile-values,profile-reorder-functions,tracer")
#pragma GCC optimize("vpt")
#pragma GCC optimize("rename-registers")
#pragma GCC optimize("move-loop-invariants")
#pragma GCC optimize("unswitch-loops")
#pragma GCC optimize("function-sections")
#pragma GCC optimize("data-sections")
#pragma GCC optimize("branch-target-load-optimize")
#pragma GCC optimize("branch-target-load-optimize2")
#pragma GCC optimize("btr-bb-exclusive")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
*/
```

```
// #define int long long
using namespace std;
typedef long double ld;
typedef long long ll;
const int inf = 1e9 + 7;
const ll l_inf = (ll) inf * (ll) inf;
const ld eps = 1e-6;


bool f(char ch) {
    return ch == '(' || ch == ')';
}


signed main() {
#ifdef LOCAL
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
#else
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#endif
    int n;
    cin >> n;
    n *= 2;
    string S;
    cin >> S;
    vector<bool> A;
    for (int i = 0; i < n; ++i) {
        if (!A.empty() && A.back() == f(S[i])) {
            A.pop_back();
        } else {
            A.push_back(f(S[i]));
        }
    }
    //
    cout << A.size() / 2 << "\n";
    cerr << 1.0 * clock() / CLOCKS_PER_SEC << "\n";
    return 0;
}
```

Task E ()

```
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <vector>
#include <set>
#include <bitset>
#include <map>
#include <string>
#include <ctime>
#include <queue>
#include <numeric>
#include <random>
#include <immintrin.h>
/*
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.2,avx,popcnt,abm,mmx,tune=native")
#pragma GCC target("avx2")
#pragma GCC optimize("Ofast,fast-math,no-stack-protector,unroll-loops,inline")
*/
/*
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.1,sse4.2,popcnt,abm,mmx,avx,avx2,tune=native")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("profile-values,profile-reorder-functions,tracer")
#pragma GCC optimize("vpt")
#pragma GCC optimize("rename-registers")
#pragma GCC optimize("move-loop-invariants")
#pragma GCC optimize("unswitch-loops")
#pragma GCC optimize("function-sections")
#pragma GCC optimize("data-sections")
#pragma GCC optimize("branch-target-load-optimize")
#pragma GCC optimize("branch-target-load-optimize2")
#pragma GCC optimize("btr-bb-exclusive")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
*/
```

```
// #define int long long
using namespace std;
typedef long double ld;
typedef long long ll;
const int inf = 1e9 + 7;
const ll l_inf = (ll) inf * (ll) inf;
const ld eps = 1e-6;
mt19937 rnd(239);


/*
int f1(int n, int k, vector<int> A) {
    mt19937 rnd(239);

} */


vector<int> deltas(1e5);


void setup1() {
    for (int i = 0; i < deltas.size(); ++i) {
        deltas[i] = rnd();
    }
}


const int N = 10, K = 3;


vector<int> f(vector<int> a) {
    sort(a.begin(), a.end());
    return a;
}


map<vector<int>, vector<vector<int>>> G;
set<vector<int>> used;
map<vector<int>, vector<int>> match;


bool dfs(vector<int> v) {
    used.insert(v);
    for (auto u : G[v]) {
        if (!match.count(u) || (used.find(match[u]) == used.end() && dfs(match[u]))) {
            match[u] = v;
            return true;
        }
    }
    return false;
}


map<vector<int>, vector<int>> match_;


void setup2() {
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < i; ++j) {
            for (int z = 0; z < j; ++z) {
                for (int l = 0; l < N; ++l) {
                    if (l != i && l != j && l != z) {
                        G[f({i, j, z})].push_back(f({i, j, z, l}));
                    }
                }
            }
        }
    }
    //
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < i; ++j) {
            for (int z = 0; z < j; ++z) {
                used.clear();
                dfs(f({i, j, z}));
            }
        }
    }
    //
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < i; ++j) {
            for (int z = 0; z < j; ++z) {
                for (int l = 0; l < z; ++l) {
                    match_[match[f({i, j, z, l})]] = f({i, j, z, l});
                }
            }
        }
    }
}


int n, k;


void solve11() {
    vector<int> A(k);
    for (int i = 0; i < k; ++i) {
        cin >> A[i];
        --A[i];
    }
    ll s = accumulate(A.begin(), A.end(), 0) % n;
    for (auto delta : deltas) {
        delta %= n;
        bool ok = true;
        for (int i = 0; i < k; ++i) {
            if (A[i] == (s + delta) % n || A[i] == (s + delta + n / 2) % n) {
                ok = false;
                break;
            }
        }
        if (ok) {
            cout << (s + delta) % n + 1 << "\n";
            break;
        }
    }
}


void solve12() {
    vector<int> A(k);
    for (int i = 0; i < k; ++i) {
        cin >> A[i];
        --A[i];
    }
    vector<int> B = match_[f(A)];
    for (int i = 0; i < k + 1; ++i) {
        if (find(A.begin(), A.end(), B[i]) == A.end()) {
            cout << B[i] + 1 << "\n";
            break;
        }
    }
}


void solve21() {
    vector<int> A(k + 1);
    for (int i = 0; i < k + 1; ++i) {
        cin >> A[i];
        --A[i];
    }
    ll S = accumulate(A.begin(), A.end(), 0) % n;
    for (auto delta : deltas) {
        delta %= n;
        vector<int> ok;
        for (int i = 0; i < k + 1; ++i) {
            if (A[i] == (S + delta) / 2 % n || A[i] == (S + delta + n) / 2 % n) {
                ok.push_back(A[i]);
            }
        }
        //
        if (ok.size() == 1) {
            for (int i = 0; i < k + 1; ++i) {
                if (A[i] != ok.front()) {
                    cout << A[i] + 1 << "␣";
                }
            }
            cout << "\n";
            break;
        }
    }
}


void solve22() {
    vector<int> A(k + 1);
    for (int i = 0; i < k + 1; ++i) {
        cin >> A[i];
        --A[i];
    }
    vector<int> B = match[f(A)];
    for (int el : B) {
        cout << el + 1 << "␣";
    }
    cout << "\n";
}


signed main() {
#ifdef LOCAL
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
#else
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#endif
    setup1();
    setup2();
    string type;
    cin >> type;
    if (type == "add") {
        int t;
        cin >> t;
        for (int _ = 0; _ < t; ++_) {
            cin >> n >> k;
            if (n == 10) {
                solve12();
            } else {
                solve11();
            }
        }
    } else {
        int t;
        cin >> t;
        for (int _ = 0; _ < t; ++_) {
            cin >> n >> k;
            if (n == 10) {
                solve22();
            } else {
                solve21();
            }
        }
    }
    cerr << 1.0 * clock() / CLOCKS_PER_SEC << "\n";
    return 0;
}
```

Task F ()

```
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <vector>
#include <set>
#include <bitset>
#include <map>
#include <string>
#include <ctime>
#include <queue>
#include <numeric>
#include <random>
#include <immintrin.h>
/*
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.2,avx,popcnt,abm,mmx,tune=native")
#pragma GCC target("avx2")
#pragma GCC optimize("Ofast,fast-math,no-stack-protector,unroll-loops,inline")
*/
/*
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.1,sse4.2,popcnt,abm,mmx,avx,avx2,tune=native")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("profile-values,profile-reorder-functions,tracer")
#pragma GCC optimize("vpt")
#pragma GCC optimize("rename-registers")
#pragma GCC optimize("move-loop-invariants")
#pragma GCC optimize("unswitch-loops")
#pragma GCC optimize("function-sections")
#pragma GCC optimize("data-sections")
#pragma GCC optimize("branch-target-load-optimize")
#pragma GCC optimize("branch-target-load-optimize2")
#pragma GCC optimize("btr-bb-exclusive")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
*/
```

```
// #define int long long
using namespace std;
typedef long double ld;
typedef long long ll;
const int inf = 1e9 + 7;
const ll l_inf = (ll) inf * (ll) inf;
const ld eps = 1e-6;


vector<pair<int, int>> movements = {{-1, 0}, {-1, 1}, {0, 1}, {1, 1},
                                    {1, 0}, {1, -1}, {0, -1}, {-1, -1}};


signed main() {
#ifdef LOCAL
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
#else
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#endif
    int n;
    cin >> n;
    cout << "4\n";
    cout << "0␣0\n␣0␣1\n␣1␣1\n␣1␣0\n";
    //
    for (int i = 0; i < n; ++i) {
        cout << movements[i].first << "␣" << movements[i].second << "\n";
    }

    cerr << 1.0 * clock() / CLOCKS_PER_SEC << "\n";
    return 0;
}
```