Олимпиада СПбГУ по информатике 2021/22 учебного года

A	B	C	D	E	F	Sum
100	100	100	80	58	31	469

Task A ()

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

using namespace std;

#define int long long

int get_st(int n) {
    int cnt = 0;
    while (n % 10 == 0) {
        cnt++;
        n /= 10;
    }
    return cnt;
}

int bin_pow(int a, int p) {
    if (p == 0) {
        return 1;
    }
    if (p % 2 == 0) {
        int k = bin_pow(a, p / 2);
        return k * k;
    }
    return a * bin_pow(a, p - 1);
}

int32_t main() {
    int n;
    cin >> n;
    int summ = 0, st = 0;
    vector<pair<int, int>> mas(n);
    for (int i = 0; i < n; i++) {
        cin >> mas[i].second >> mas[i].first;
        mas[i].first += get_st(mas[i].second);
        mas[i].second /= bin_pow(10, get_st(mas[i].second));
    }
    sort(mas.begin(), mas.end());
    summ = mas[0].second;
    st = mas[0].first;
    for (int i = 1; i < n; i++) {
        if (st < mas[i].first) {
            cout << st;
            return 0;
        } else {
            summ *= bin_pow(10, st - mas[i].first);
            summ += mas[i].second;
            st = mas[i].first + get_st(summ);
            summ /= bin_pow(10, get_st(summ));
        }
    }
    cout << st;
}
```

Task B ()

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

using namespace std;

int main() {
    int n;
    cin >> n;
    vector<int> mas(n);
    for (int i = 0; i < n; i++) {
        cin >> mas[i];
    }
    for (int i = n - 1; i >= 0; i--) {
        for (int j = 0; j < mas[i]; j++) {
            cout << "Flip and wait" << endl;
            int x = 0;
            string s;
            while (cin >> s) {
                x += (s.size() - 2) / 2;
                if (x >= i + 1) {
                    break;
                }
                cout << "Wait" << endl;
            }
        }
    }
    cout << "Stop" << endl;
}
```

Task C ()

```cpp
#include <iostream>

using namespace std;

void solve() {
    string s;
    cin >> s;
    bool is_q = 0;
    for (char i : s) {
        if (i == '?') {
            is_q = 1;
        }
    }
    string ans = "";
    if (is_q) {
        for (int i = 0; i < s.size() - 1; i += 2) {
            if (s[i] == '?' && s[i + 1] == '0') {
                ans += "00";
            } else if (s[i] == '0' && s[i + 1] == '?') {
                ans += "01";
            } else if (s[i] == '1' && s[i + 1] == '?') {
                ans += "10";
            } else {
                ans += "11";
            }
        }
        if (s.size() % 2 == 1) {
            ans += s[s.size() - 1];
        }
    } else {
        for (int i = 0; i < s.size() - 1; i += 2) {
            if (s[i] == '0' && s[i + 1] == '0') {
                ans += "?0";
            } else if (s[i] == '0' && s[i + 1] == '1') {
                ans += "0?";
            } else if (s[i] == '1' && s[i + 1] == '0') {
                ans += "1?";
            } else {
                ans += "?1";
            }
        }
        if (s.size() % 2 == 1) {
            ans += s[s.size() - 1];
        }
    }
    cout << ans << endl;;
}

int main() {
    int t;
    cin >> t;
    while (t--) solve();
}
```

Task D ()

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

using namespace std;

const int MOD = 998244353;

int add(int a, int b) {
    a += b;
    if (a < 0) a += MOD;
    if (a >= MOD) a -= MOD;
    return a;
}

int mul(long long a, int b) {
    return a * b % MOD;
}

int main() {
    int n;
    cin >> n;
    int ans = 1;
    vector<int> mas = {1, 808258749, 117153405, 761699708, 573994984, 62402409, 511621808,
        242726978, 887890124, 875880304};
    n++;
    for (int i = 0; i <= 9; i++) {
        if (n >= i * int(1e8) && n <= (i + 1) * int(1e8)) {
            ans = mas[i];
            for (int j = i * int(1e8) + 1; j <= n; j++) {
                ans = mul(ans, j);
            }
        }
    }
    cout << 0 << " " << ans << endl;
}
```

Task E ()

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

using namespace std;

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    int n;
    cin >> n;
    vector<int> v(n);
    for (int i = 0; i < n; i++) {
        cin >> v[i];
    }
    int q;
    cin >> q;
    while (q--) {
        int a, b, d;
        cin >> a >> b >> d;
        a--, b--;
        long long summ = 0;
        for (int i = a + 1; i <= b; i++) {
            summ += (d + v[i] - 1) / v[i];
        }
        cout << summ << '\n';
    }
}
```

Task F ()

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

using namespace std;

int rnd() {
    return (rand() << 15) + rand();
}

struct Node {
    long long x;
    int y, val, mtree;
    Node *l, *r;
    Node() {
        x = 0, y = rnd();
        val = 0;
        mtree = 0;
        l = r = nullptr;
    }
    Node(long long _x, int _val) {
        x = _x;
        val = _val;
        mtree = val;
        y = rnd();
        l = r = nullptr;
    }
};

int get_mtree(Node* root) {
    if (!root) {
        return 0;
    }
    return root->mtree;
}

void upd(Node* root) {
    if (!root) {
        return;
    }
    root->mtree = max(max(get_mtree(root->l), get_mtree(root->r)), root->val);
}

pair<Node*, Node*> split(Node* root, long long x) {
    if (!root) return {nullptr, nullptr};
    if (root->x <= x) {
        auto t = split(root->r, x);
        root->r = t.first;
        upd(root);
        return {root, t.second};
    } else {
        auto t = split(root->l, x);
        root->l = t.second;
        upd(root);
        return {t.first, root};
    }
}

Node* merge(Node* left, Node* right) {
    if (!left) return right;
    if (!right) return left;
    if (left->y > right->y) {
        left->r = merge(left->r, right);
        upd(left);
        return left;
    } else {
        right->l = merge(left, right->l);
        upd(right);
        return right;
    }
}
```

```
struct segtree {
    vector<Node*> t;
    segtree() {
        t.resize(1e6, nullptr);
    }
    void add_val(int v, int vl, int vr, int prefl, long long prefr, int val) {
        auto p = split(t[v], prefr);
        auto p1 = split(p.first, prefr - 1);
        if (!p1.second) {
            t[v] = merge(p1.first, new Node(prefr, val));
            t[v] = merge(t[v], p.second);
        } else {
            p1.second->val = max(p1.second->val, val);
            upd(p1.second);
            t[v] = merge(p1.first, new Node(prefr, val));
            t[v] = merge(t[v], p1.second);
            t[v] = merge(t[v], p.second);
        }
        if (vl == vr - 1) {
            return;
        }
        int vm = (vl + vr) / 2;
        if (prefl < vm) {
            add_val(2 * v, vl, vm, prefl, prefr, val);
        } else {
            add_val(2 * v + 1, vm, vr, prefl, prefr, val);
        }
    }
    long long get_maxx(int v, int vl, int vr, int prefl, int kon, long long prefr) {
        if (vl >= kon || vr <= prefl) {
            return 0;
        }
        if (vl >= prefl && vr <= kon) {
            auto p = split(t[v], prefr);
            int ans = get_mtree(p.first);
            t[v] = merge(p.first, p.second);
            return ans;
        }
        int vm = (vl + vr) / 2;
        return max(get_maxx(2 * v, vl, vm, prefl, kon, prefr), get_maxx(2 * v + 1, vm, vr, prefl,
            kon, prefr));
    }
};

int32_t main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    srand(time(0));
    int n;
    cin >> n;
    vector<pair<int, int>> mas(n);
    vector<long long> prefl(n + 1), prefr(n + 1);
    for (int i = 0; i < n; i++) {
        cin >> mas[i].first >> mas[i].second;
    }
    for (int i = 1; i <= n; i++) {
        prefl[i] = prefl[i - 1] + mas[i - 1].first;
        prefr[i] = prefr[i - 1] + mas[i - 1].second;
    }
    vector<long long> sprefl = prefl;
    sort(sprefl.begin(), sprefl.end());
    sprefl.resize(unique(sprefl.begin(), sprefl.end()) - sprefl.begin());
    int m = sprefl.size();
    vector<int> dp(n + 1);
    segtree tree;
    dp[0] = 1;
    tree.add_val(1, 0, m, lower_bound(sprefl.begin(), sprefl.end(), 0) - sprefl.begin(), 0, 1);
    for (int i = 1; i <= n; i++) {
        int val = tree.get_maxx(1, 0, m, lower_bound(sprefl.begin(), sprefl.end(), prefl[i]) -
            sprefl.begin(), m, prefr[i]);
        dp[i] = val + 1;
        tree.add_val(1, 0, m, lower_bound(sprefl.begin(), sprefl.end(), prefl[i]) - sprefl.begin()
```

```
            ,  prefr[i], dp[i]);
    }
    cout << *max_element(dp.begin(), dp.end());
}
```