Олимпиада СПбГУ по информатике 2022/23 учебного года

A	B	C	D	E	F	Sum
100	100	90	60	52	65	467

Task A ()

```
//#pragma GCC optimize("Ofast")
//#pragma GCC target("avx2")

#include <bits/stdc++.h>

#define int long long

using namespace std;

typedef long long ll;

const int mod = 2286661337;


void solve() {
    int n;
    //cin >> n;
    n = 6;
    vector<int> a;
    for (int i = 0; i < n; ++i) {
        //cin >> a[i];
        int x;
        cin >> x;
        x--;
        a.insert(a.begin() + x, i);
    }
    vector<int> difficulties(n);
    for (int i = 0; i < n; ++i) {
        //cout << a[i] << ' ';
        difficulties[a[i]] = i + 1;
    }
    for (int i = 0; i < n; ++i) {
        cout << difficulties[i] << '␣';
    }
}


signed main() {
    //ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

Task B ()

```cpp
//#pragma GCC optimize("Ofast")
//#pragma GCC target("avx2")

#include <bits/stdc++.h>

#define int long long

using namespace std;

typedef long long ll;

const int mod = 2286661337;


void solve() {
    string run;
    cin >> run;
    if (run == "first") {
        int n;
        cin >> n;
        int sum = 0;
        for (int i = 0; i < n; ++i) {
            int x;
            cin >> x;
            sum += x;
        }
        sum *= 100000000000000ll;
        cout << sum;
    } else {
        int n;
        cin >> n;
        int sum = 0;
        for (int i = 0; i < n; ++i) {
            int x;
            cin >> x;
            if (i == 0) {
                sum = x / 1000000000000000ll;
            }
            sum += (x % 100000000000000ll);
        }
        cout << sum;
    }
}


signed main() {
    //ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

Task C ()

```
//#pragma GCC optimize("Ofast")
//#pragma GCC target("avx2")

#include <bits/stdc++.h>

#define int long long

using namespace std;

typedef long long ll;

const int mod = 2286661337;


void solve() {
    array<int, 3> a{}, b{};
    set<pair<int, int>> rectangles;
    for (int i = 0; i < 3; ++i) {
        cin >> a[i] >> b[i];
        if (a[i] > b[i])
            swap(a[i], b[i]);
        rectangles.insert({a[i], b[i]});
    }
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            if (j == i)
                continue;
            if (a[i] == a[j] && b[i] > b[j]) {
                pair<int, int> new_rectangle = {a[i], b[i] - b[j]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
            }
            if (a[i] > a[j] && b[i] > b[j]) {
                for (int k = 0; k < 3; ++k) {
                    if (k == i || k == j)
                        continue;
                    if (b[i] - b[j] == b[k] && a[j] == a[k]) {
                        pair<int, int> new_rectangle = {a[i] - a[j], b[i]};
                        if (new_rectangle.first > new_rectangle.second)
                            swap(new_rectangle.first, new_rectangle.second);
                        if (new_rectangle.first > 0)
                            rectangles.insert(new_rectangle);
                    }
                    if (b[k] <= b[i] - b[j] && a[k] > a[i] - a[j]) {
                        pair<int, int> new_rectangle = {a[i] - a[j], b[j]};
                        if (new_rectangle.first > new_rectangle.second)
                            swap(new_rectangle.first, new_rectangle.second);
                        rectangles.insert(new_rectangle);
                    }
                    if (b[k] < b[i] - b[j] && a[k] == a[i]) {
                        pair<int, int> new_rectangle = {a[i], b[i] - b[j] - b[k]};
                        if (new_rectangle.first > new_rectangle.second)
                            swap(new_rectangle.first, new_rectangle.second);
                        if (new_rectangle.first > 0)
                            rectangles.insert(new_rectangle);
                    }
                    swap(a[k], b[k]);
                    if (b[i] - b[j] == b[k] && a[j] == a[k]) {
                        pair<int, int> new_rectangle = {a[i] - a[j], b[i]};
                        if (new_rectangle.first > new_rectangle.second)
                            swap(new_rectangle.first, new_rectangle.second);
                        if (new_rectangle.first > 0)
                            rectangles.insert(new_rectangle);
                    }
                    if (b[k] <= b[i] - b[j] && a[k] > a[i] - a[j]) {
                        pair<int, int> new_rectangle = {a[i] - a[j], b[j]};
                        if (new_rectangle.first > new_rectangle.second)
                            swap(new_rectangle.first, new_rectangle.second);
                        if (new_rectangle.first > 0)
                            rectangles.insert(new_rectangle);
```

```
            }
            if (b[k] < b[i] - b[j] && a[k] == a[i]) {
                pair<int, int> new_rectangle = {a[i], b[i] - b[j] - b[k]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
            }
            swap(a[k], b[k]);
        }
    }
    swap(a[j], b[j]);
    if (a[i] == a[j] && b[i] > b[j]) {
        pair<int, int> new_rectangle = {a[i], b[i] - b[j]};
        if (new_rectangle.first > new_rectangle.second)
            swap(new_rectangle.first, new_rectangle.second);
        if (new_rectangle.first > 0)
            rectangles.insert(new_rectangle);
    }
    if (a[i] > a[j] && b[i] > b[j]) {
        for (int k = 0; k < 3; ++k) {
            if (k == i || k == j)
                continue;
            if (b[i] - b[j] == b[k] && a[j] == a[k]) {
                pair<int, int> new_rectangle = {a[i] - a[j], b[i]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
            }
            if (b[k] <= b[i] - b[j] && a[k] > a[i] - a[j]) {
                pair<int, int> new_rectangle = {a[i] - a[j], b[j]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
            }
            if (b[k] < b[i] - b[j] && a[k] == a[i]) {
                pair<int, int> new_rectangle = {a[i], b[i] - b[j] - b[k]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
            }
            if (a[k] + a[j] == a[i] && b[j] + b[k] == b[i]) {
                pair<int, int> new_rectangle = {a[j], b[k]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
                new_rectangle = {a[k], b[j]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
            }
            swap(a[k], b[k]);
            if (b[i] - b[j] == b[k] && a[j] == a[k]) {
                pair<int, int> new_rectangle = {a[i] - a[j], b[i]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
            }
            if (a[k] + a[j] == a[i] && b[j] + b[k] == b[i]) {
                pair<int, int> new_rectangle = {a[j], b[k]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
                new_rectangle = {a[k], b[j]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
```

```
                    rectangles.insert(new_rectangle);
                }
                if (b[k] <= b[i] - b[j] && a[k] > a[i] - a[j]) {
                    pair<int, int> new_rectangle = {a[i] - a[j], b[j]};
                    if (new_rectangle.first > new_rectangle.second)
                        swap(new_rectangle.first, new_rectangle.second);
                    if (new_rectangle.first > 0)
                        rectangles.insert(new_rectangle);
                }
                if (b[k] < b[i] - b[j] && a[k] == a[i]) {
                    pair<int, int> new_rectangle = {a[i], b[i] - b[j] - b[k]};
                    if (new_rectangle.first > new_rectangle.second)
                        swap(new_rectangle.first, new_rectangle.second);
                    if (new_rectangle.first > 0)
                        rectangles.insert(new_rectangle);
                }
                swap(a[k], b[k]);
            }
        }
        swap(a[j], b[j]);
    }
    swap(a[i], b[i]);
    for (int j = 0; j < 3; ++j) {
        if (j == i)
            continue;
        if (a[i] == a[j] && b[i] > b[j]) {
            pair<int, int> new_rectangle = {a[i], b[i] - b[j]};
            if (new_rectangle.first > new_rectangle.second)
                swap(new_rectangle.first, new_rectangle.second);
            if (new_rectangle.first > 0)
                rectangles.insert(new_rectangle);
        }
        if (a[i] > a[j] && b[i] > b[j]) {
            for (int k = 0; k < 3; ++k) {
                if (k == i || k == j)
                    continue;
                if (b[i] - b[j] == b[k] && a[j] == a[k]) {
                    pair<int, int> new_rectangle = {a[i] - a[j], b[i]};
                    if (new_rectangle.first > new_rectangle.second)
                        swap(new_rectangle.first, new_rectangle.second);
                    if (new_rectangle.first > 0)
                        rectangles.insert(new_rectangle);
                }
                if (b[k] <= b[i] - b[j] && a[k] > a[i] - a[j]) {
                    pair<int, int> new_rectangle = {a[i] - a[j], b[j]};
                    if (new_rectangle.first > new_rectangle.second)
                        swap(new_rectangle.first, new_rectangle.second);
                    rectangles.insert(new_rectangle);
                }
                if (a[k] + a[j] == a[i] && b[j] + b[k] == b[i]) {
                    pair<int, int> new_rectangle = {a[j], b[k]};
                    if (new_rectangle.first > new_rectangle.second)
                        swap(new_rectangle.first, new_rectangle.second);
                    if (new_rectangle.first > 0)
                        rectangles.insert(new_rectangle);
                    new_rectangle = {a[k], b[j]};
                    if (new_rectangle.first > new_rectangle.second)
                        swap(new_rectangle.first, new_rectangle.second);
                    if (new_rectangle.first > 0)
                        rectangles.insert(new_rectangle);
                }
                if (b[k] < b[i] - b[j] && a[k] == a[i]) {
                    pair<int, int> new_rectangle = {a[i], b[i] - b[j] - b[k]};
                    if (new_rectangle.first > new_rectangle.second)
                        swap(new_rectangle.first, new_rectangle.second);
                    if (new_rectangle.first > 0)
                        rectangles.insert(new_rectangle);
                }
                swap(a[k], b[k]);
                if (b[i] - b[j] == b[k] && a[j] == a[k]) {
                    pair<int, int> new_rectangle = {a[i] - a[j], b[i]};
                    if (new_rectangle.first > new_rectangle.second)
                        swap(new_rectangle.first, new_rectangle.second);
                    if (new_rectangle.first > 0)
```

```
                rectangles.insert(new_rectangle);
            }
            if (a[k] + a[j] == a[i] && b[j] + b[k] == b[i]) {
                pair<int, int> new_rectangle = {a[j], b[k]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
                new_rectangle = {a[k], b[j]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
            }
            if (b[k] <= b[i] - b[j] && a[k] > a[i] - a[j]) {
                pair<int, int> new_rectangle = {a[i] - a[j], b[j]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
            }
            if (b[k] < b[i] - b[j] && a[k] == a[i]) {
                pair<int, int> new_rectangle = {a[i], b[i] - b[j] - b[k]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
            }
            swap(a[k], b[k]);
        }
    }
    swap(a[j], b[j]);
    if (a[i] == a[j] && b[i] > b[j]) {
        pair<int, int> new_rectangle = {a[i], b[i] - b[j]};
        if (new_rectangle.first > new_rectangle.second)
            swap(new_rectangle.first, new_rectangle.second);
        if (new_rectangle.first > 0)
            rectangles.insert(new_rectangle);
    }
    if (a[i] > a[j] && b[i] > b[j]) {
        for (int k = 0; k < 3; ++k) {
            if (k == i || k == j)
                continue;
            if (b[i] - b[j] == b[k] && a[j] == a[k]) {
                pair<int, int> new_rectangle = {a[i] - a[j], b[i]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
            }
            if (b[k] <= b[i] - b[j] && a[k] > a[i] - a[j]) {
                pair<int, int> new_rectangle = {a[i] - a[j], b[j]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
            }
            if (a[k] + a[j] == a[i] && b[j] + b[k] == b[i]) {
                pair<int, int> new_rectangle = {a[j], b[k]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
                new_rectangle = {a[k], b[j]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
                    rectangles.insert(new_rectangle);
            }
            if (b[k] < b[i] - b[j] && a[k] == a[i]) {
                pair<int, int> new_rectangle = {a[i], b[i] - b[j] - b[k]};
                if (new_rectangle.first > new_rectangle.second)
                    swap(new_rectangle.first, new_rectangle.second);
                if (new_rectangle.first > 0)
```

```
                        rectangles.insert(new_rectangle);
                    }
                    swap(a[k], b[k]);
                    if (b[i] - b[j] == b[k] && a[j] == a[k]) {
                        pair<int, int> new_rectangle = {a[i] - a[j], b[i]};
                        if (new_rectangle.first > new_rectangle.second)
                            swap(new_rectangle.first, new_rectangle.second);
                        if (new_rectangle.first > 0)
                            rectangles.insert(new_rectangle);
                    }
                    if (b[k] <= b[i] - b[j] && a[k] > a[i] - a[j]) {
                        pair<int, int> new_rectangle = {a[i] - a[j], b[j]};
                        if (new_rectangle.first > new_rectangle.second)
                            swap(new_rectangle.first, new_rectangle.second);
                        if (new_rectangle.first > 0)
                            rectangles.insert(new_rectangle);
                    }
                    if (a[k] + a[j] == a[i] && b[j] + b[k] == b[i]) {
                        pair<int, int> new_rectangle = {a[j], b[k]};
                        if (new_rectangle.first > new_rectangle.second)
                            swap(new_rectangle.first, new_rectangle.second);
                        if (new_rectangle.first > 0)
                            rectangles.insert(new_rectangle);
                        new_rectangle = {a[k], b[j]};
                        if (new_rectangle.first > new_rectangle.second)
                            swap(new_rectangle.first, new_rectangle.second);
                        if (new_rectangle.first > 0)
                            rectangles.insert(new_rectangle);
                    }
                    if (b[k] < b[i] - b[j] && a[k] == a[i]) {
                        pair<int, int> new_rectangle = {a[i], b[i] - b[j] - b[k]};
                        if (new_rectangle.first > new_rectangle.second)
                            swap(new_rectangle.first, new_rectangle.second);
                        if (new_rectangle.first > 0)
                            rectangles.insert(new_rectangle);
                    }
                    swap(a[k], b[k]);
                }
            }
            swap(a[j], b[j]);
        }
        swap(a[i], b[i]);
    }
    for (auto& rec: rectangles) {
        cout << rec.first << ' ' << rec.second << '\n';
    }
}


signed main() {
    //ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

Task D ()

```
//#pragma GCC optimize("Ofast")
//#pragma GCC target("avx2")

#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>

#define int long long

using namespace std;
using namespace __gnu_pbds;

typedef long long ll;

const int mod = 2286661337;
const int p = 127;

int calc_hash(vector<pair<int, char>> now_pos) {
    pair<int, int> hash = {0, 0};
    for (int i = 0; i < now_pos.size(); ++i) {
        hash.first *= p;
        hash.first %= mod;
        hash.first += now_pos[i].first;
        hash.first %= mod;
        hash.second *= 7;
        hash.second %= mod;
        hash.second += now_pos[i].second;
        hash.first %= mod;
    }
    int hhash = hash.first * 997 + hash.second;
    hhash %= mod;
    return hhash;
}

void retroanalisys(vector<pair<int, char>> now_pos, gp_hash_table<int, char>& is_win, vector<int>&
    a) {
    int sum = 0;
    if (is_win[calc_hash(now_pos)]) {
        return;
    }
    is_win[calc_hash(now_pos)] = 1;
    for (int i = 0; i < now_pos.size(); ++i) {
        sum += now_pos[i].first + now_pos[i].second;
    }
    if (sum == 0) {
        return;
    }
    bool flag = 0;
    for (int i = 0; i < now_pos.size(); ++i) {
        if (now_pos[i].second) {
            now_pos[i].second = 0;
            int prev_val = now_pos[i].first;
            now_pos[i].first = a[i];
            retroanalisys(now_pos, is_win, a);
            if (is_win[calc_hash(now_pos)] == 1) {
                now_pos[i] = {prev_val, 1};
                is_win[calc_hash(now_pos)] = 2;
                return;
            }
            now_pos[i] = {prev_val, 1};
        }
        for (int j = 1; j <= now_pos[i].first; ++j) {
            now_pos[i].first -= j;
            retroanalisys(now_pos, is_win, a);
            if (is_win[calc_hash(now_pos)] == 1) {
                now_pos[i].first += j;
                is_win[calc_hash(now_pos)] = 2;
                return;
            }
            now_pos[i].first += j;
        }
    }
}
```

```
void solve() {
    int n;
    cin >> n;
    vector<int> a(n);
    int Xor = 0;
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
        Xor ^= a[i];
    }
    /*if (n == 1) {
        cout << 1 << ' ' << a[0] << endl;
        int x, y;
        cin >> x >> y;
        if (x == -1)
            return;
        cout << 1 << ' ' << a[0] << endl;
        cin >> x >> y;
        if (x == -1)
            return;
    }*/
    vector<pair<int, char>> now_pos(n);
    for (int i = 0; i < n; ++i) {
        now_pos[i] = {a[i], 1};
    }
    gp_hash_table<int, char> is_win;
    retroanalisys(now_pos, is_win, a);
    if (is_win[calc_hash(now_pos)] == 1) {
        cout << "-1 -1" << endl;
    } else {
        while (1) {
            /*int sum = 0;
            for (int i = 0; i < now_pos.size(); ++i) {
                sum += now_pos[i].first + now_pos[i].second;
            }
            if (sum == 0) {
                //is_win[now_pos] = 1;
                cout << "-1 -1" << endl;
                return;
            }*/
            //bool flag = 0;
            for (int i = 0; i < now_pos.size(); ++i) {
                if (now_pos[i].second) {
                    now_pos[i].second = 0;
                    int prev_val = now_pos[i].first;
                    now_pos[i].first = a[i];
                    //retroanalisys(now_pos, is_win);
                    if (is_win[calc_hash(now_pos)] == 1) {
                        //now_pos[i].second = 1;
                        //is_win[now_pos] = 2;
                        //return;
                        cout << i + 1 << ' ' << 0 << endl;
                        break;
                    }
                    now_pos[i] = {prev_val, 1};
                }
                for (int j = 1; j <= now_pos[i].first; ++j) {
                    now_pos[i].first -= j;
                    //retroanalisys(now_pos, is_win);
                    if (is_win[calc_hash(now_pos)] == 1) {
                        cout << i + 1 << ' ' << j << endl;
                        break;
                        /*now_pos[i].first += j;
                        is_win[now_pos] = 2;
                        return;*/
                    }
                    now_pos[i].first += j;
                }
            }
            int mobius_pos, mobius_val;
            cin >> mobius_pos >> mobius_val;
            if (mobius_pos == -1) {
                return;
```

```
            } else {
                if (mobius_val == 0) {
                    now_pos[mobius_pos - 1].first = a[mobius_pos - 1];
                    now_pos[mobius_pos - 1].second = 0;
                } else
                    now_pos[mobius_pos - 1].first -= mobius_val;
            }
        }
    }
}


signed main() {
    //ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

Task E ()

```
//#pragma GCC optimize("Ofast")
//#pragma GCC target("avx2")

#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>

#define int long long

using namespace std;
using namespace __gnu_pbds;

typedef long long ll;


void solve() {
    int n;
    cin >> n;
    string run;
    cin >> run;
    if (run == "transmit") {
        for (int t = 0; t < n; ++t) {
            int x;
            cin >> x;
            vector<vector<char>> ans(10, vector<char>(10, '0'));
            //int log = __lg(x);
            for (int i = 0; i < 10; ++i) {
                if (x == 0)
                    break;
                int to_decrease = 1;
                ans[i][0]++;
                for (int j = 1; j < 10; ++j) {
                    if (to_decrease * 2 <= x) {
                        ans[i][j]++;
                        to_decrease <<= 1;
                    } else
                        break;
                }
                x -= to_decrease;
            }
            for (int i = 0; i < 10; ++i) {
                for (int j = 0; j < 10; ++j) {
                    cout << ans[i][j];
                }
                cout << '\n';
            }
        }
    } else {
        for (int t = 0; t < n; ++t) {
            vector<string> table(10);
            vector<int> degrees(10);
            int ans = 0;
            for (int i = 0; i < 10; ++i) {
                cin >> table[i];
                for (int j = 0; j < 10; ++j) {
                    degrees[i] += table[i][j] - '0';
                }
                if (degrees[i] > 0) {
                    ans += (1ll<<(degrees[i] - 1));
                }
            }
            cout << ans << '\n';
        }
    }
}


signed main() {
    //ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    int t = 1;
    //cin >> t;
    while (t--) {
```

```
        solve();
    }
    return 0;
}
```

Task F ()

```
a = input()
b = input()
#print(a[1:5])
#print(b[4:10])
new_a = ''
new_b = ''
opening_bracket = -1
for i in range(len(a)):
    if a[i] == ')':
        tmp = a[opening_bracket + 1:i]
        ttmp = list(tmp.split('|'))
        new_a += ttmp[0] * int(ttmp[1])
        opening_bracket = -1
    elif a[i] == '(':
        opening_bracket = i
    elif opening_bracket == -1:
        new_a += a[i]
opening_bracket = -1
for i in range(len(b)):
    if b[i] == ')':
        tmp = b[opening_bracket + 1:i]
        ttmp = list(tmp.split('|'))
        new_b += ttmp[0] * int(ttmp[1])
        opening_bracket = -1
    elif b[i] == '(':
        opening_bracket = i
    elif opening_bracket == -1:
        new_b += b[i]
print(int(new_a) + int(new_b))
```