Main Template

```cpp
#include <bits/stdc++.h>
using namespace std;

#define endl '\n'
#define getVec(arr, size)   \
    vector<int> arr(size);  \
    for (auto &input : arr) cin >> input;

#define print(z, n)                                           \
    for (int i = 0; (n && i < n) || (!n && i < z.size()); i++) \
        cout << z[i] << ' ';                                  \
    cout << endl;

#define FIO { ios_base::sync_with_stdio(false); cin.tie(nullptr); cout.tie(nullptr); }

const int M = 1e9 + 7, OO = 1e9;

int dx[] = {1, -1, 0, 0, 1, 1, -1, -1};
int dy[] = {0, 0, 1, -1, -1, 1, -1, 1};
string dd[] = {"U", "D", "R", "L", "UL", "UR", "DL", "DR"};

void solve(){

}

signed main()
{
    FIO

    #ifndef ONLINE_JUDGE
        freopen("input.txt", "r", stdin);
        freopen("output.txt", "w", stdout);
    #endif

    int t = 1;
    // cin >> t;
    for (int i = 1; i <= t; i++)
    {
        solve();
        cout << endl;
    }
    // cerr << clock() / 1000.0 << " Secs";
}

// ####################
// ##### 3Bcaren0 #####
// ####################
```

Ordered set

```cpp
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
#define ordered_set tree<int, null_type,less<>, rb_tree_tag,tree_order_statistics_node_update>// set

typedef tree<int, null_type,less_equal<int>, rb_tree_tag,tree_order_statistics_node_update> ordered_multiset;
```

Interactive and Receive

Print __int128

```cpp
int answer(vector<int> have){

    cout << "? ";
    for(auto &x: have) cout << x << ' ';
    cout << endl;

    cout.flush();

    int ans; cin >> ans;
    return ans;
}
```

```cpp
void print_int128(__int128 n) {
    if (n == 0) {
        cout << "0";
        return;
    }
    string s;
    while (n > 0) {
        s += '0' + (n % 10);
        n /= 10;
    }
    reverse(s.begin(), s.end());
    cout << s;
}
```

Random

```cpp
std::mt19937_64 rnd(std::chrono::system_clock::now().time_since_epoch().count());

    int l = LLONG_MIN, r = LLONG_MAX;
    // Generate a random number between l and r
    uniform_int_distribution<int> dist(l, r);
    int random_number = dist(rnd);
```