```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <numeric>
#include <complex>

using namespace std;


constexpr int X_MAX = 17630;
constexpr int Y_MAX = 9000;
constexpr int R = 5000;
constexpr int RR = R * R;


class entity{
     public:
          int id; // Unique identifier
          int type; // 0=monster, 1=your hero, 2=opponent hero
          int x; // Position of this entity
          int y;
          int shield_life; // Ignore for this league; Count down until
shield spell fades
          int is_controlled; // Ignore for this league; Equals 1 when
this entity is under a control spell
          int health; // Remaining health of this monster
          int vx; // Trajectory of this monster
          int vy;
          int near_base; // 0=monster with no target yet, 1=monster
targeting a base
          int threat_for; // Given this monster's trajectory, is it a
threat to 1=your base, 2=your opponent's base, 0=neither
        int rr;
        int err;
        int i;
};
int dist(int x, int y) {
    return x * x + y * y;
}
int dist(int x, int y, int ox, int oy) {
    return dist(x - ox, y - oy);
}
int dist(entity e, entity h) {
    return dist(e.x - h.x, e.y - h.y);
}
pair<int, int> position(int x, int y, int ox, int oy, int r) {
    int u = x - ox, v = x - oy;
    int d = dist(u, v);
    int rr = r * r;
    double rd = r;
```

```
    if (d <= r) {
        return {x, y};
    }
    complex<double> cuv(u, v), c = polar(rd, arg(cuv));
    ox += c.real();
    oy += c.imag();
    return {ox, oy};
}

pair<bool, entity> closest(const vector<entity>& ms, int x, int y, int
d) {
    int rr = dist(x - ms[0].x, y - ms[0].y), v;
    int dd = d * d;
    entity e = ms[0];
    for (const auto& m: ms) {
        v = dist(x - m.x, y - m.y);
        if (rr < v) {
            rr = v;
            e = m;
        }
    }
    bool b = rr <= dd;
    return {b, e};
}
pair<bool, entity> closest(const vector<entity>& ms, entity e, int d) {
    return closest(ms, e.x, e.y, d);
}

int main()
{
     int base_x, base_ex = X_MAX; // The corner of the map
representing your base
     int base_y, base_ey = Y_MAX;
    int ux = X_MAX - 2000, uy = 0;
    int vx = X_MAX, vy = Y_MAX - 2000;
     cin >> base_x >> base_y; cin.ignore();
    if (0 != base_x) {
        base_ex = base_ey = 0;
        ux = 2000;
        vx = vy = 0;
    }
    int DD = dist(base_x - base_ex,  base_y - base_ey);
     int heroes_per_player; // Always 3
     cin >> heroes_per_player; cin.ignore();
    vector<int> hero(heroes_per_player), enemy(heroes_per_player);
    iota(begin(hero), end(hero), 0);
     vector<entity> my_monsters;
     vector<entity> my_heros;
     vector<entity> enemy_heros;
     vector<entity> enemy_monsters;
     vector<entity> neutral_monsters;
     // game loop
```

```
    long long step = 0;
    pair<bool, entity> eb, h0, h1, h2;
    vector<entity> hs(heroes_per_player);
    bool wind1, wind2;
    int x, y;
     while (1) {
        wind1 = wind2 = true;
           int health, ehealth; // Your base health
           int mana, emana; // Ignore in the first league; Spend ten
mana to cast a spell
           for (int i = 0, a, b; i < 2; i++) {
                if (0 == i) {
                cin >> health >> mana; cin.ignore();
            } else {
                cin >> ehealth >> emana; cin.ignore();
            }
           }
           int entity_count; // Amount of heros and monsters you can
see
           cin >> entity_count; cin.ignore();
           entity e, h;
           for (int i = 0; i < entity_count; i++) {
                cin >> e.id >> e.type >> e.x >> e.y >> e.shield_life >>
e.is_controlled >> e.health >> e.vx >> e.vy >> e.near_base >>
e.threat_for; cin.ignore();
            e.rr = dist(e.x - base_x, e.y - base_y);
            e.err = dist(e.x - base_ex, e.y - base_ey);
                if (e.type == 1)
                     my_heros.push_back(e);
            if (e.type == 2)
                     enemy_heros.push_back(e);
            if (e.type == 0 && e.threat_for == 0)
                     neutral_monsters.push_back(e);
            if (e.type == 0 && e.threat_for == 1)
                     my_monsters.push_back(e);
            if (e.type == 0 && e.threat_for == 2)
                     enemy_monsters.push_back(e);
           }
        sort(begin(hero), end(hero), [&](auto lhs, auto rhs){
                return my_heros[lhs].rr < my_heros[rhs].rr;
                });
        sort(begin(enemy_heros), end(enemy_heros), [&](auto lhs, auto
rhs){
                return lhs.rr < rhs.rr;
                });
        sort(begin(my_monsters), end(my_monsters), [&](auto lhs, auto
rhs){
                return lhs.rr < rhs.rr;
                });
        sort(begin(enemy_monsters), end(enemy_monsters), [&](auto lhs,
auto rhs){
                return lhs.rr < rhs.rr;
```

```
                });
        sort(begin(neutral_monsters), end(neutral_monsters), [&](auto
lhs, auto rhs){
                return lhs.rr < rhs.rr;
                });
        ++step;
           for (int i = 0; i < heroes_per_player; i++) {
            my_heros[hero[i]].i = i;
            hs[i] = my_heros[hero[i]];
        }
                // if (60 < mana && !enemy_heros.empty() && (eb =
closest(my_monsters, enemy_heros.back())).second) {
           for (int i = 0; i < heroes_per_player; i++) {
            h = my_heros[i];
            if (0 == my_heros[i].i) {
                if (step < 5) {
                    cout << "WAIT" << endl;
                } else if (my_monsters.empty()) {
                    if (neutral_monsters.empty()) {
                        cout << "MOVE " << base_ex << " " << base_ey <<
endl;
                    } else {
                        e = neutral_monsters[0];
                        if (dist(e, h) <= 4000000 && 30 < mana) {
                            mana -= 10;
                            cout << "SPELL CONTROL " << e.id << " " <<
base_ex << " " << base_ey << endl;
                        } else {
                            cout << "MOVE " << neutral_monsters[0].x <<
" " << neutral_monsters[0].y << endl;
                        }
                    }
                } else if (my_monsters[0].rr <= 490000 && 10 <= mana) {
                    mana -= 10;
                    cout << "SPELL WIND " <<  base_ex << " " << base_ey
<< endl;
                } else {
                    cout << "MOVE " << my_monsters[0].x << " " <<
my_monsters[0].y << endl;
                }
            } else if (1 == my_heros[i].i) {
                if (step < 5) {
                    cout << "WAIT" << endl;
                } else if (!enemy_heros.empty() && 5 *
enemy_heros[0].rr < enemy_heros[0].err && 2 < my_monsters.size()) {
                    e = enemy_heros[0];
                    if (30 < mana && dist(e, h) <= 4000000) {
                        mana -= 10;
                        if (e.err < e.rr) {
                            cout << "SPELL CONTROL " << e.id << " " <<
base_x << " " << base_y << endl;
                        } else {
```

```
                            cout << "SPELL CONTROL " << e.id << " " <<
base_ex << " " << base_ey << endl;
                        }
                    } else {
                        x = (hs[0].x + e.x) / 2;
                        y = (hs[0].y + e.y) / 2;
                        cout << "MOVE " << x << " " << y << endl;
                    }
                }  else if (40 < mana && !enemy_monsters.empty() && (eb
= closest(enemy_monsters, h, 2000)).first) {
                    e = eb.second;
                    if (30 < mana) {
                        mana -= 10;
                        if (wind1) {
                            wind1 = true;
                            cout << "SPELL WIND " << base_ex << " " <<
base_ey << endl;
                        } else {
                            cout << "SPELL SHIELD " << e.id << " " <<
hs[1].x << " " << hs[1].y << endl;
                        }
                    } else {
                        cout << "MOVE " << e.x + e.vx << " " << e.y +
e.vy << endl;
                    }
                } else if ( neutral_monsters.empty()) {
                    if (h.rr + 5000000 < h.err) {
                        cout << "MOVE " << ux << " " << uy << endl;
                    } else {
                        cout << "MOVE " << base_x << " " << base_y <<
endl;
                    }
                } else {
                    eb = closest(neutral_monsters, h, 2000);
                    e = eb.second;
                    if (eb.first) {
                        if (30 < mana) {
                            mana -= 10;
                            if (h.rr < e.rr) {
                                cout << "SPELL WIND " << base_ex << " "
<< base_ey << endl;
                            } else {
                                cout << "SPELL CONTROL " << e.id << " "
<< base_ex << " " << base_ey << endl;
                            }
                        } else if (10 < mana && e.rr <= 490000) {
                            mana -= 10;
                            cout << "SPELL WIND " << base_ex << " " <<
base_ey << endl;
                        } else {
                            cout << "WAIT" << endl;
                        }
```

```
                    } else {
                        cout << "MOVE " << e.x + e.vx << " " << e.y +
e.vy << endl;
                    }
                }
            } else {
                if (30 < mana && !enemy_monsters.empty()) {
                    eb = closest(enemy_monsters, h, 2000);
                    e = eb.second;
                    if (eb.first) {
                        if (30 < mana) {
                            mana -= 10;
                            if (wind2 && e.err < h.err) {
                                wind2 = false;
                                cout << "SPELL WIND " << base_ex << " "
<< base_ey << endl;
                            } else {
                                cout << "SPELL SHIELD " << e.id <<
endl;
                            }
                        } else {
                            cout << "WAIT" << endl;
                        }
                    } else {
                        cout << "MOVE " << e.x + e.vx << " " << e.y +
e.vy << endl;
                    }
                } else if (30 < mana && !enemy_heros.empty() && (eb =
closest(enemy_heros, h, 2000)).first) {
                    mana -= 10;
                    e = eb.second;
                    if (e.err < e.rr) {
                            cout << "SPELL CONTROL " << e.id << " " <<
base_x << " " << base_y << endl;
                        } else {
                            cout << "SPELL CONTROL " << e.id << " " <<
base_ex << " " << base_ey << endl;
                        }
                } else if (my_monsters.size() < 12 &&
neutral_monsters.empty()) {
                    if (20000000 < h.err) {
                        cout << "MOVE " << base_ex << " " << base_ey <<
endl;
                    } else {
                        cout << "MOVE " << base_x << " " << base_y <<
endl;
                    }
                } else if (12 <= my_monsters.size()) {
                    eb = closest(my_monsters, h, 2000);
                    e = eb.second;
                    if (eb.first) {
                        if (30 < mana) {
```

```
                            mana -= 10;
                            cout << "SPELL CONTROL " << e.id << " " <<
base_ex << " " << base_ey << endl;
                        } else {
                            cout << "WAIT" << endl;
                        }
                    } else {
                        if (5000000 < h.err) {
                            cout << "MOVE " << base_ex << " " <<
base_ey << endl;
                        } else {
                            cout << "MOVE " << base_x << " " << base_y
<< endl;
                        }
                    }
                } else {
                    eb = closest(neutral_monsters, h, 2000);
                    e = eb.second;
                    if (eb.first) {
                        if (30 < mana) {
                            mana -= 10;
                            cout << "SPELL CONTROL " << e.id << " " <<
base_ex << " " << base_ey << endl;
                        } else if (10 < mana && e.rr <= 490000) {
                            mana -= 10;
                            cout << "SPELL WIND " << " " << base_ex <<
" " << base_ey << endl;
                        } else {
                            cout << "WAIT" << endl;
                        }
                    } else {
                        if (20000000 < h.err) {
                            cout << "MOVE " << base_ex << " " <<
base_ey << endl;
                        } else {
                            cout << "MOVE " << base_x << " " << base_y
<< endl;
                        }
                    }
                }
             }
           }

           my_monsters.clear();
           enemy_monsters.clear();
           my_heros.clear();
           enemy_heros.clear();
           neutral_monsters.clear();

     }
}
```