Problem

There are a lot of great streetside food vendors in Manhattan, but without a doubt, the one with the tastiest food is th
Code Jam Crepe Cart!

You want to find the cart, but you do not know where it is, except that it is at some street intersection. You believe t
at people from across Manhattan are currently walking toward that intersection, so you will try to identify the interse
tion toward which the most people are traveling.

For the purposes of this problem, Manhattan is a regular grid with its axes aligned to compass lines and bounded bet
een 0 and Q, inclusive, on each axis. There are west-east streets corresponding to gridlines y = 0, y = 1, y = 2, …, y
Q and south-north streets corresponding to gridlines x = 0, x = 1, x = 2, …, x = Q, and people move only along thes
streets. The points where the lines meet — e.g., (0, 0) and (1, 2) — are intersections. The shortest distance between
wo intersections is measured via the Manhattan distance — that is, by the sum of the absolute horizontal difference a
d the absolute vertical difference between the two sets of coordinates.

You know the locations of P people, all of whom are standing at intersections, and the compass direction each perso
is headed: north (increasing y direction), south (decreasing y direction), east (increasing x direction), or west (decre
sing x direction). A person is moving toward a street intersection if their current movement is on a shortest path to th
t street intersection within the Manhattan grid. For example, if a person located at (x0, y0) is moving north, then the
are moving toward all street intersections that have coordinates (x, y) with y > y0.

You think the crepe cart is at the intersection toward which the most people are traveling. Moreover, you believe tha
more southern and western parts of the island are most likely to have a crepe cart, so if there are multiple such inters
ctions, you will choose the one with the smallest non-negative x coordinate, and if there are multiple such intersectio
s with that same x coordinate, the one among those with the smallest non-negative y coordinate. Which intersection
ill you choose?

Input

The first line of the input gives the number of test cases, T. T test cases follow. Each test case starts with one line co
taining two integers P and Q: the number of people, and the maximum possible value of an x or y coordinate in Man
attan, as described above. Then, there are P more lines. The i-th of those lines contains two integers Xi and Yi, the c
rrent location (street corner) of a person, and a character Di, the direction that person is headed. Di is one of the uppe
case letters N, S, E, or W, which stand for north, south, east, and west, respectively.

Output

For each test case, output one line containing Case #t: x y, where t is the test case number (starting from 1) and x an
y are the horizontal and vertical coordinates of the intersection where you believe the crepe cart is located.

Limits

$1 \leq T \leq 100$.
Time limit: 20 seconds per test set.
Memory limit: 1GB.
$1 \leq P \leq 500$.
$0 \leq Xi \leq Q$, for all i.
$0 \leq Yi \leq Q$, for all i.
For all i, if $Xi = 0$, $Di \neq W$.
For all i, if $Yi = 0$, $Di \neq S$.
For all i, if $Xi = Q$, $Di \neq E$.
For all i, if $Yi = Q$, $Di \neq N$.

Test set 1 (Visible)
$Q = 10$.

Test set 2 (Hidden)
Q = 105.

Sample

Input

Output

```
3
1 10
5 5 N
4 10
2 4 N
2 6 S
1 5 E
3 5 W
8 10
0 2 S
0 3 N
0 3 N
0 4 N
0 5 S
0 5 S
0 8 S
1 5 W
```

```
Case #1: 0 6
Case #2: 2 5
Case #3: 0 4
```

In Sample Case #1, there is only one person, and they are moving north from (5, 5). This means that all street corner with y ≥ 6 are possible locations for the crepe cart. Of those possibilities, we choose the one with lowest x ≥ 0 and t en lowest y ≥ 6.

In Sample Case #2, there are four people, all moving toward location (2, 5). There is no other location that has as ma y people moving toward it.

In Sample Case #3, six of the eight people are moving toward location (0, 4). There is no other location that has as any people moving toward it.

Solution:

```
#pragma GCC optimize ("O3")
#pragma GCC target ("sse4")

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

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/rope>

using namespace std;
using namespace __gnu_pbds;
using namespace __gnu_cxx;

typedef long long ll;
typedef long double ld;
typedef complex<ld> cd;

typedef pair<int, int> pi;
typedef pair<ll,ll> pl;
typedef pair<ld,ld> pd;

typedef vector<int> vi;
typedef vector<ld> vd;
typedef vector<ll> vl;
typedef vector<pi> vpi;
typedef vector<pl> vpl;
typedef vector<cd> vcd;

template <class T> using Tree = tree<T, null_type, less<T>, rb_tree_tag,tree_order_statistics_node_update>;

#define FOR(i, a, b) for (int i = (a); i < (b); i++)
#define F0R(i, a) for (int i = 0; i < (a); i++)
#define FORd(i,a,b) for (int i = (b)-1; i >= (a); i--)
#define F0Rd(i,a) for (int i = (a)-1; i >= 0; i--)
#define trav(a, x) for (auto& a : x)

#define mp make_pair
#define pb push_back
#define f first
#define s second
#define lb lower_bound
#define ub upper_bound

#define sz(x) (int)x.size()
#define beg(x) x.begin()
#define en(x) x.end()
#define all(x) beg(x), en(x)
#define resz resize

const int MOD = 1000000007; // 998244353
const ll INF = 1e18;
const int MX = 100005;
const ld PI = 4*atan((ld)1);

template<class T> void ckmin(T &a, T b) { a = min(a, b); }
template<class T> void ckmax(T &a, T b) { a = max(a, b); }

namespace input {
    template<class T> void re(complex<T>& x);
    template<class T1, class T2> void re(pair<T1,T2>& p);
    template<class T> void re(vector<T>& a);
    template<class T, size_t SZ> void re(array<T,SZ>& a);

    template<class T> void re(T& x) { cin >> x; }
    void re(double& x) { string t; re(t); x = stod(t); }
    void re(ld& x) { string t; re(t); x = stold(t); }
    template<class Arg, class... Args> void re(Arg& first, Args&... rest) {
        re(first); re(rest...);
    }

    template<class T> void re(complex<T>& x) { T a,b; re(a,b); x = cd(a,b); }
    template<class T1, class T2> void re(pair<T1,T2>& p) { re(p.f,p.s); }
    template<class T> void re(vector<T>& a) { F0R(i,sz(a)) re(a[i]); }
    template<class T, size_t SZ> void re(array<T,SZ>& a) { F0R(i,SZ) re(a[i]); }
}

using namespace input;

namespace output {
    template<class T1, class T2> void pr(const pair<T1,T2>& x);
    template<class T, size_t SZ> void pr(const array<T,SZ>& x);
    template<class T> void pr(const vector<T>& x);
    template<class T> void pr(const set<T>& x);
    template<class T1, class T2> void pr(const map<T1,T2>& x);

    template<class T> void pr(const T& x) { cout << x; }
    template<class Arg, class... Args> void pr(const Arg& first, const Args&... rest) {
        pr(first); pr(rest...);
    }

    template<class T1, class T2> void pr(const pair<T1,T2>& x) {
        pr("{",x.f,", ",x.s,"}");
    }
    template<class T> void prContain(const T& x) {
        pr("{");
        bool fst = 1; trav(a,x) pr(!fst?", ":"",a), fst = 0;
        pr("}");
    }
    template<class T, size_t SZ> void pr(const array<T,SZ>& x) { prContain(x); }
    template<class T> void pr(const vector<T>& x) { prContain(x); }
    template<class T> void pr(const set<T>& x) { prContain(x); }
    template<class T1, class T2> void pr(const map<T1,T2>& x) { prContain(x); }

    void ps() { pr("\n"); }
    template<class Arg, class... Args> void ps(const Arg& first, const Args&... rest) {
        pr(first," "); ps(rest...); // print w/ spaces
    }
}

using namespace output;
```

```
namespace io {
    void setIn(string s) { freopen(s.c_str(),"r",stdin); }
    void setOut(string s) { freopen(s.c_str(),"w",stdout); }
    void setIO(string s = "") {
        ios_base::sync_with_stdio(0); cin.tie(0); // fast I/O
        if (sz(s)) { setIn(s+".in"), setOut(s+".out"); } // for USACO
    }
}

using namespace io;

template<class T> T invGeneral(T a, T b) {
    a %= b; if (a == 0) return b == 1 ? 0 : -1;
    T x = invGeneral(b,a);
    return x == -1 ? -1 : ((1-(ll)b*x)/a+b)%b;
}

template<class T> struct modular {
    T val;
    explicit operator T() const { return val; }
    modular() { val = 0; }
    template<class U> modular(const U& v) {
        val = (-MOD <= v && v <= MOD) ? v : v % MOD;
        if (val < 0) val += MOD;
    }
    friend ostream& operator<<(ostream& os, const modular& a) { return os << a.val; }
    friend bool operator==(const modular& a, const modular& b) { return a.val == b.val; }
    friend bool operator!=(const modular& a, const modular& b) { return !(a == b); }

    modular operator-() const { return modular(-val); }
    modular& operator+=(const modular& m) { if ((val += m.val) >= MOD) val -= MOD; return *this; }
    modular& operator-=(const modular& m) { if ((val -= m.val) < 0) val += MOD; return *this; }
    modular& operator*=(const modular& m) { val = (ll)val*m.val%MOD; return *this; }
    friend modular exp(modular a, ll p) {
        modular ans = 1; for (; p; p /= 2, a *= a) if (p&1) ans *= a;
        return ans;
    }
    friend modular inv(const modular& a) { return invGeneral(a.val,MOD); }
    // inv is equivalent to return exp(b,b.mod-2) if prime
    modular& operator/=(const modular& m) { return (*this) *= inv(m); }

    friend modular operator+(modular a, const modular& b) { return a += b; }
    friend modular operator-(modular a, const modular& b) { return a -= b; }
    friend modular operator*(modular a, const modular& b) { return a *= b; }

    friend modular operator/(modular a, const modular& b) { return a /= b; }
};

typedef modular<int> mi;
typedef pair<mi,mi> pmi;
typedef vector<mi> vmi;
typedef vector<pmi> vpmi;

int P,Q, cum[2][MX];

void solve(int caseNum) {
    re(P,Q);
    F0R(i,Q+3) cum[0][i] = cum[1][i] = 0;
    F0R(i,P) {
        int X, Y; char D; re(X,Y,D);
        if (D == 'N') {
            cum[1][Y+1] ++;
            cum[1][Q+1] --;
        }
        if (D == 'S') {
            cum[1][0] ++;
            cum[1][Y] --;
        }
        if (D == 'W') {
            cum[0][0] ++;
            cum[0][X] --;
        }
        if (D == 'E') {
            cum[0][X+1] ++;
            cum[0][Q+1] --;
        }
    }
    pi bes[2] = {{-MOD,-MOD},{-MOD,-MOD}};
    F0R(i,Q+1) {
        if (i) cum[0][i] += cum[0][i-1], cum[1][i] += cum[1][i-1];
        F0R(j,2) if (cum[j][i] > bes[j].f) bes[j] = {cum[j][i],i};
    }
    ps(bes[0].s,bes[1].s);
    // cerr << "Solved #" << caseNum << "\n";
}

int main() {
    setIO();
    int T; re(T);
    FOR(i,1,T+1) {
        pr("Case #",i,": ");
        solve(i);
    }
}

/* stuff you should look for
    * int overflow, array bounds
    * special cases (n=1?), set tle
    * do smth instead of nothing and stay organized
*/
```