Problem
Banny has just bought a new programmable robot. Eager to test his coding skills, he has placed the robot in a grid of squares with R rows (numbered 1 to R from north to south) and C columns (numbered 1 to C from west to east). The square in row r and column c is denoted (r, c).

Initially the robot starts in the square (SR, SC). Banny will give the robot N instructions. Each instruction is one of , S, E or W, instructing the robot to move one square north, south, east or west respectively.

If the robot moves into a square that it has been in before, the robot will continue moving in the same direction until t reaches a square that it has not been in before. Banny will never give the robot an instruction that will cause it to m ve out of the grid.

Can you help Banny determine which square the robot will finish in, after following the N instructions?

Input
The first line of the input gives the number of test cases, T. T test cases follow. Each test case starts with a line conta ning the five integers N, R, C, SR and SC, the number of instructions, the number of rows, the number of columns, t e robot's starting row and starting column, respectively.

Then, another line follows containing a single string of N characters; the i-th of these characters is the i-th instructio Banny gives the robot (one of N, S, E or W, as described above).

Output
For each test case, output one line containing Case #x: r c, where x is the test case number (starting from 1), r is the r w the robot finishes in and c is the column the robot finishes in.

Limits
Memory limit: 1GB.
$1 \leq T \leq 100$.
$1 \leq R \leq 5 \times 104$.
$1 \leq C \leq 5 \times 104$.
$1 \leq SR \leq R$.
$1 \leq SC \leq C$.
The instructions will not cause the robot to move out of the grid.

Test set 1 (Visible)
Time limit: 20 seconds.
$1 \leq N \leq 100$.

Test set 2 (Hidden)
Time limit: 60 seconds.
$1 \leq N \leq 5 \times 104$.

Sample

Input

Output

3
5 3 6 2 3
EEWNS

4 3 3 1 1
SESE
11 5 8 3 4
NEESSWWNESE

Case #1: 3 2
Case #2: 3 3
Case #3: 3 7

Sample Case #1 corresponds to the top-left diagram, Sample Case #2 corresponds to the top-right diagram and Samp e Case #3 corresponds to the lower diagram. In each diagram, the yellow square is the square the robot starts in, whil the green square is the square the robot finishes in.

```
#include <set>
#include <string>
#include <vector>
#include <iostream>
using namespace std;
int N; string s; vector<set<pair<int, int> > > sx, sy;
void add(int x, int y) {
 set<pair<int, int> >::iterator it1 = sx[y].lower_bound(make_pair(x + 1, 0));
 set<pair<int, int> >::iterator it2 = it1;
 if (it2 == sx[y].begin()) it2 = sx[y].end();
 else it2--;
 if (it1 != sx[y].end() && it1->first == x + 1) {
  if (it2 != sx[y].end() && it2->second == x) {
   sx[y].insert(make_pair(it2->first, it1->second));
   sx[y].erase(it2);
  }
  else sx[y].insert(make_pair(x, it1->second));
  sx[y].erase(it1);
 }
 else if (it2 != sx[y].end() && it2->second == x) {
  sx[y].insert(make_pair(it2->first, x + 1));
  sx[y].erase(it2);
 }
 else sx[y].insert(make_pair(x, x + 1));
 it1 = sy[x].lower_bound(make_pair(y + 1, 0));
 it2 = it1;
 if (it2 == sy[x].begin()) it2 = sy[x].end();
 else it2--;
 if (it1 != sy[x].end() && it1->first == y + 1) {
  if (it2 != sy[x].end() && it2->second == y) {
   sy[x].insert(make_pair(it2->first, it1->second));
   sy[x].erase(it2);
  }
  else sy[x].insert(make_pair(y, it1->second));
  sy[x].erase(it1);
 }
```

```
 else if (it2 != sy[x].end() && it2->second == y) {
  sy[x].insert(make_pair(it2->first, y + 1));
  sy[x].erase(it2);
 }
 else sy[x].insert(make_pair(y, y + 1));
}
int main() {
 int Q;
 cin >> Q;
 for (int rep = 1; rep <= Q; ++rep) {
  int R, C, sr, sc;
  cin >> N >> R >> C >> sr >> sc >> s;
  int x = N, y = N;
  sx.clear();
  sy.clear();
  sx.resize(2 * N + 1); sx[N].insert(make_pair(N, N + 1));
  sy.resize(2 * N + 1); sy[N].insert(make_pair(N, N + 1));
  for (int i = 0; i < N; i++) {
   if (s[i] == 'N' || s[i] == 'S') {
    set<pair<int, int> >::iterator it = --sx[y].lower_bound(make_pair(x + 1, 0));
    if (s[i] == 'S') x = it->second;
    if (s[i] == 'N') x = it->first - 1;
   }
   if (s[i] == 'E' || s[i] == 'W') {
    set<pair<int, int> >::iterator it = --sy[x].lower_bound(make_pair(y + 1, 0));
    if (s[i] == 'E') y = it->second;
    if (s[i] == 'W') y = it->first - 1;
   }
   add(x, y);
  }
  cout << "Case #" << rep << ": " << x - N + sr << ' ' << y - N + sc << endl;
 }
 return 0;
}
```