Problem

Marlin is a fish who lost his son and is trying to find him. Fortunately, he ran into Cynthia, a turtle, as she swam aro nd with her brothers, Wally and Seymour. Cynthia knows exactly where Marlin needs to go, and she can be very pre ise in giving directions. While Marlin is smart and can follow them perfectly, keeping track of a long list of direction can be problematic. Cynthia needs to find a way to make the list of directions short.

Marlin lives in a matrix of R rows and C columns. Some cells of the matrix are dangerous and cannot be entered. M rlin and his son are currently in different non-dangerous cells. Marlin's son never moves to a different cell. Cynthia d cided to give Marlin directions in the form of a program consisting of a list of instructions, each on a single line. Eac instruction is of one of 5 types:

N: move one cell North (up),
S: move one cell South (down),
W: move one cell West (left),
E: move one cell East (right), and
G(i): jump to the i-th line of the instruction list (counting starting from 1).
After executing a line with any of the first 4 instructions, Marlin jumps to the next line on the list if there is one. If th re is no next line, Marlin just stands still forever.

For example, if Marlin were following the program

1: N
2: E
3: G(6)
4: S
5: G(1)
6: W
7: G(4)
he would move North (line 1), then East (2), then jump to line 6 without physically moving (3), then move West (6), then jump to line 4 (7), then move South (4), then jump to line 1 (5), then move North (1), etc.

If at any point Marlin and his son are at the same cell, they will be reunited and Marlin will no longer follow any inst uctions. Cynthia the turtle wants to find out the smallest number of lines in a program that would get Marlin to the s me cell as his son, without him ever going into a dangerous cell or moving outside of the matrix boundaries. All G i structions must jump to existing lines in the program.

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 R and C, the number of rows and columns in the matrix. Then, R lines follow containing a string of C character each. The j-th character of the i-th of these lines Aij represents the cell in the i-th row and j-th column of the matrix. The character is # if the cell is dangerous, an uppercase M if the cell is the one Marlin is currently at, an uppercase N if the cell is the one Marlin's son is currently at and . if the cell is an unoccupied non-dangerous cell.

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is I POSSIBLE if there is no program that would get Marlin to his son under the conditions explained above, or the smal est number of instructions in such a program.

Limits

Memory limit: 1GB.
$1 \leq T \leq 100$.
Aij is either #, ., uppercase M or uppercase N, for all i and j.

Aij = M for exactly one pair of i and j.
Aij = N for exactly one pair of i and j.

Test set 1 (Visible)
Time limit: 30 seconds.
$1 \leq R \leq 10$.
$1 \leq C \leq 10$.

Test set 2 (Hidden)
Time limit: 120 seconds.

For at most 10 test cases:
$1 \leq R \leq 100$.
$1 \leq C \leq 100$.

For the remaining test cases:
$1 \leq R \leq 50$.
$1 \leq C \leq 50$.

Sample

Input

Output

```
5
2 5
N...#
....M
2 5
N#...
...#M
5 5
N..##
#.###
#...#
##.##
##..M
5 5
..N##
#.###
#...#
##.##
##..M
3 3
#M#
###
#N#
```

Case #1: 4
Case #2: 7

Case #3: 5
Case #4: 6
Case #5: IMPOSSIBLE

Below are some shortest programs for each of the possible sample case.

Sample Case #1:
1: W
2: N
3: S
4: G(1)

or
1: W
2: N
3: W
4: G(3)

.
Sample Case #2:
1: N
2: W
3: W
4: S
5: W
6: W
7: N

.
Sample Case #3:
1: W
2: W
3: N
4: N
5: G(2)

.
Sample Case #4:
1: W
2: W
3: N
4: N
5: E
6: G(1)

.
Notice that even though the program must contain the smallest possible number of lines, it is not required to minimi e the number of moves that Marlin makes.

Solution:

```
#include <bits/stdc++.h>

using namespace std;

template <typename A, typename B>
string to_string(pair<A, B> p);

template <typename A, typename B, typename C>
string to_string(tuple<A, B, C> p);

template <typename A, typename B, typename C, typename D>
string to_string(tuple<A, B, C, D> p);

string to_string(const string& s) {
  return "" + s + "";
}

string to_string(const char* s) {
  return to_string((string) s);
}

string to_string(bool b) {
  return (b ? "true" : "false");
}

string to_string(vector<bool> v) {
  bool first = true;
  string res = "{";
  for (int i = 0; i < static_cast<int>(v.size()); i++) {
    if (!first) {
      res += ", ";
    }
    first = false;
    res += to_string(v[i]);
  }
  res += "}";
  return res;
}

template <size_t N>
string to_string(bitset<N> v) {
  string res = "";
  for (size_t i = 0; i < N; i++) {
    res += static_cast<char>('0' + v[i]);
  }
  return res;
}

template <typename A>
string to_string(A v) {
  bool first = true;
  string res = "{";
```

```
  for (const auto &x : v) {
    if (!first) {
      res += ", ";
    }
    first = false;
    res += to_string(x);
  }
  res += "}";
  return res;
}

template <typename A, typename B>
string to_string(pair<A, B> p) {
  return "(" + to_string(p.first) + ", " + to_string(p.second) + ")";
}

template <typename A, typename B, typename C>
string to_string(tuple<A, B, C> p) {
  return "(" + to_string(get<0>(p)) + ", " + to_string(get<1>(p)) + ", " + to_string(get<2>(p)) + ")";
}

template <typename A, typename B, typename C, typename D>
string to_string(tuple<A, B, C, D> p) {
  return "(" + to_string(get<0>(p)) + ", " + to_string(get<1>(p)) + ", " + to_string(get<2>(p)) + ", " + to_string(get<3
(p)) + ")";
}

void debug_out() { cerr << endl; }

template <typename Head, typename... Tail>
void debug_out(Head H, Tail... T) {
  cerr << " " << to_string(H);
  debug_out(T...);
}

#ifdef LOCAL
#define debug(...) cerr << "[" << #__VA_ARGS__ << "]:", debug_out(__VA_ARGS__)
#else
#define debug(...) 42
#endif

const int DX[4] = {-1, 0, 1, 0};
const int DY[4] = {0, -1, 0, 1};

const int MAX = 40010;

int qx[MAX];
int qy[MAX];

void RunBfs(const vector<string>& s, int sx, int sy, vector<vector<int>>& dist) {
  int h = (int) s.size();
  int w = (int) s[0].size();
```

```
  for (int i = 0; i < h; i++) {
    for (int j = 0; j < w; j++) {
      dist[i][j] = -1;
    }
  }
  if (s[sx][sy] == '#') {
    return;
  }
  int b = 0, e = 1;
  qx[0] = sx;
  qy[0] = sy;
  dist[sx][sy] = 0;
  while (b < e) {
    for (int dir = 0; dir < 4; dir++) {
      int nx = qx[b] + DX[dir];
      int ny = qy[b] + DY[dir];
      if (nx >= 0 && ny >= 0 && nx < h && ny < w) {
        if (s[nx][ny] != '#' && dist[nx][ny] == -1) {
          qx[e] = nx;
          qy[e] = ny;
          dist[nx][ny] = dist[qx[b]][qy[b]] + 1;
          ++e;
        }
      }
    }
    ++b;
  }
}

int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  cout << fixed << setprecision(17);
  int tt;
  cin >> tt;
  for (int qq = 1; qq <= tt; qq++) {
    cout << "Case #" << qq << ": ";
    int h, w;
    cin >> h >> w;
    vector<string> s(h);
    for (int i = 0; i < h; i++) {
      cin >> s[i];
    }
    int sx = -1, sy = -1;
    int fx = -1, fy = -1;
    for (int i = 0; i < h; i++) {
      for (int j = 0; j < w; j++) {
        if (s[i][j] == 'M') {
          sx = i;
          sy = j;
        }
        if (s[i][j] == 'N') {
```

```
      fx = i;
      fy = j;
    }
  }
}
vector<vector<int>> dist(h, vector<int>(w));
RunBfs(s, sx, sy, dist);
vector<vector<int>> dist_start = dist;
if (dist[fx][fy] == -1) {
  cout << "IMPOSSIBLE" << '\n';
  continue;
}
int ans = dist[fx][fy];
RunBfs(s, fx, fy, dist);
vector<vector<int>> dist_finish = dist;
for (int dx = -h + 1; dx <= h - 1; dx++) {
  for (int dy = -w + 1; dy <= w - 1; dy++) {
    if (dx == 0 && dy == 0) {
      continue;
    }
    int mx = fx - dx;
    int my = fy - dy;
    if (mx < 0 || my < 0 || mx >= h || my >= w || s[mx][my] == '#') {
      continue;
    }
    RunBfs(s, mx, my, dist);
    vector<vector<int>> dist_old = dist;
    vector<string> g = s;
    for (int cycles = 1; ; cycles++) {
      for (int i = 0; i < h; i++) {
        for (int j = 0; j < w; j++) {
          if (g[i][j] != '#') {
            int ni = i - dx * cycles;
            int nj = j - dy * cycles;
            if (ni < 0 || nj < 0 || ni >= h || nj >= w || s[ni][nj] == '#') {
              g[i][j] = '#';
            }
          }
        }
      }
      if (g[fx][fy] == '#') {
        break;
      }
      RunBfs(g, fx, fy, dist);
      for (int zx = 0; zx < h; zx++) {
        for (int zy = 0; zy < w; zy++) {
          if (g[zx][zy] == '#') {
            continue;
          }
          int ux = zx - dx * cycles;
          int uy = zy - dy * cycles;
          assert(ux >= 0 && uy >= 0 && ux < h && uy < w);
```

```
            if (dist[zx][zy] == -1 || dist_old[zx][zy] == -1 || dist_start[ux][uy] == -1) {
              continue;
            }
            int cur = dist_start[ux][uy] + dist[zx][zy] + dist_old[zx][zy] + 1;
//            debug(dx, dy, cycles, zx, zy, cur, g, dist_old[zx][zy]);
            ans = min(ans, cur);
          }
        }
        RunBfs(g, mx, my, dist);
        dist_old = dist;
//        debug(dx, dy, cycles, g, mx, my, dist);
      }
    }
  }
  cout << ans << '\n';
 }
 return 0;
}
```