```
/* Problem 2--Invisibility
   This technique is actually crude, but this was the last program I
   wrote, and I was tired.  We keep track of each square we visit and
   the remaining allowable invisibility when we get there. */

import java.io.*;
import java.util.*;

public class prob2 {

 private static Scanner in = null;
 private static PrintWriter out = null;
 private static int cs = 0;
 private static int[] rpos=null,cpos=null,invs=null,ct=null;
 private static boolean[][][] V = null;
 private static char[][] mz = null;

 public static void main (String[] args) throws Exception {

  in = new Scanner (new File ("prob2.in"));
  out = new PrintWriter ("prob2.out");
  while (true) { //read in the maze
   int r = in.nextInt (), c = in.nextInt ();
   if (r==0 && c==0) break;
   in.nextLine();
   mz = new char[r][];
   for (int i=0; i < r; i++) mz[i] = in.nextLine().toCharArray();
   int inv = in.nextInt ();
   Process (r,c,inv); //process the maze
  }
  in.close ();
  out.close ();
 }

 /* Process the maze */
 public static void Process (int r, int c, int inv) {

  rpos = new int[r*c*(inv+1)]; cpos = new int[r*c*(inv+1)];
  invs = new int[r*c*(inv+1)]; ct = new int[r*c*(inv+1)];
  //Use arrays as crude queues to store row, col, invisibilities, and
  //moves
  V = new boolean[r][c][inv+1]; //keep track of visited
  int startr = 0, startc = 0;
  loop1: for (startr=0; startr < r; startr++)
   for (startc=0; startc < c; startc++) //locate Frodo
    if (mz[startr][startc]=='F') break loop1;
   mz[startr][startc] = ' ';
  rpos[0] = startr; cpos[0] = startc; invs[0] = inv; ct[0] = 0;
  V[startr][startc][inv] = true;
  int len = 1, p = 0;
  while (p < len) { //check next square to visit
   int pr = rpos[p], pc=cpos[p], pi=invs[p], pct = ct[p]; p++;
   if (mz[pr][pc]=='D') { //did we get there?
    out.println ("Case "+(++cs)+": Frodo can get to Mount Doom in "+
                 pct+" moves.");
    out.println ();
    return;
   }
   V[pr][pc][pi] = true; //Visit, and add neighbors
   len = AddLoc (pr+1,pc,pi,pct,len);
   len = AddLoc (pr-1,pc,pi,pct,len);
   len = AddLoc (pr,pc+1,pi,pct,len);
   len = AddLoc (pr,pc-1,pi,pct,len);
  } //If loop ends, we didn't get there
  out.println ("Case "+(++cs)+": Frodo cannot get to Mount Doom.");
  out.println ();
 }

```

```
 69   /* Method that tests appropriate places to move */
 70   public static int AddLoc (int pr, int pc, int inv, int pct, int len) {
 71
 72    //If legitimate and unvisited square
 73    if ((mz[pr][pc]=='D' || mz[pr][pc]==' ') && !V[pr][pc][inv]) {
 74     if (OrcSafe (pr,pc)) { //If the square is Orc Safe, great!
 75      rpos[len] = pr; cpos[len] = pc; invs[len] = inv; ct[len] = pct+1;
 76      len++; V[pr][pc][inv] = true;
 77     } else if (inv > 0) { //Otherwise make sure I can turn invisible
 78      rpos[len] = pr; cpos[len] = pc; invs[len] = inv-1; ct[len] = pct+1;
 79      len++; V[pr][pc][inv] = true;
 80     }
 81    }
 82    return len;
 83   }
 84
 85   //OrcSafe checks all directions for Orcs
 86   public static boolean OrcSafe (int pr, int pc) {
 87
 88    int t = pr+1;
 89    while (true)
 90     if (mz[t][pc]=='O') return false;
 91     else if (mz[t][pc]=='*') break;
 92     else t++;
 93    t = pr-1;
 94    while (true)
 95     if (mz[t][pc]=='O') return false;
 96     else if (mz[t][pc]=='*') break;
 97     else t--;
 98    t = pc+1;
 99    while (true)
100     if (mz[pr][t]=='O') return false;
101     else if (mz[pr][t]=='*') break;
102     else t++;
103    t = pc-1;
104    while (true)
105     if (mz[pr][t]=='O') return false;
106     else if (mz[pr][t]=='*') break;
107     else t--;
108    return true;
109   }
110
111  }
112
```