Problem 5—Age Maze

Oh, no! Not another maze problem! Even the zombies are sick of them! However, this one is pretty easy, at least, for the zombies. All mazes are 10x10 grids. The starting point is always the NW corner; the ending point is always the SE corner. There are no boundaries, all paths to the north, south, east, and west are open, unless it would take you out of bounds of the maze. Thus, the zombie can walk freely through the maze. What's the trick? Here's the trick: Some of the borders between squares in the maze have been fixed with aging or deaging rays. When a zombie passes between squares, he will either age or deage by the number of years (positive or negative) expressed in the ray. Aging rays are unidirectional only. If a zombie passes backwards through a aging ray, the aging isn't necessarily undone, unless another aging ray is set up in the other direction, and even then, there's no guarantee that the second aging ray will undo the first. It might even make it worse! (For example, a 100-year aging ray between (3,4) and (3,5) and a 50-year aging ray between (3,5) and (3,4). A trip from (3,4) to (3,5) and back to (3,4) will age the zombie 150 years!

Look, zombies are just like everyone else. They are just as vain about their appearance as you and I, and they don't want to be any older than they have to be. So the zombie is going to go through the maze in such a way as to make him as young as possible when he reaches the destination square. How young that is, is your task. It could be that he could give himself a negative age. In that case, you don't have to compute the exact negative age, just say that he returned to the womb. It is perfectly legal for the zombie to visit a square more than once as it moves through the maze, including the destination square.

INPUT SPECIFICATION. You will be given a set of input cases, each of which will begin with an unsigned decimal integer followed by <**EOLN**>. This indicates the number of aging/deaging rays which are to follow. Each aging/deaging ray will be specified on successive lines of the format "*r* c *d a*<**EOLN**>" where *r* is the row number, *c* is the column number (both between 0 and 9 inclusive), *d* is the direction you would move from (*r*,*c*) (N,S,E,W) to pass through the age ray, and *a* is the number of years the zombie would age passing through it (or deage if the number is negative). There will be exactly one space between arguments on these lines. Finally each data case will end with an unsigned decimal integer followed by <**EOLN**>, representing the starting age of the zombie. All the aging ray specifications will be legal; in particular, there won't be multiple rays set up on any particular boundary and no aging ray will be set up on an outer age of the maze since that would take the zombie out of bounds anyway.

OUTPUT SPECIFICATION. The output cases should appear in the same order as the input cases. Each output case should be of the form "Case *c*: The zombie's final age is *a*.<**EOLN**>" where *c* is the input case, and *a* is the final age. If the zombie could achieve a negative age, the format is "Case *c*: The zombie returns to the womb.<**EOLN**>"

SAMPLE INPUT.

```
0<EOLN>
20<EOLN>
1<EOLN>
5•5•N•-1<EOLN>
20<EOLN>
-1<EOLN>
<EOF>
```

SAMPLE OUTPUT.

```
Case•1:••The•zombie's•final•age•is•20.<EOLN>
Case•2:••The•zombie•returns•to•the•womb.<EOLN>
<EOF>
```