How my program Works

Part 1

We called a second *path* predicate which has an extra parameter (the list of already visited positions). The base case for the recursion is when the initial and ending squares of the path are the same. For the recursive case we first check that N > 0 (we still have movements available) and then find a new movement. Then we check if that the square we are moving to has not been visited yet, and if so we try to find a path from that new square to the final square.

Part 2

For this problem we used the path predicate implemented in part 1. We basically find all the possible paths from A to B and we return the shortest one. So we call path with N=100, to ensure that we will find all the paths up to 10x10 boards. For each found path we check against the shortest path found so far; and if the path is shorter than the shortest path, we update the shortest path. In order to force Prolog to find all the solutions we always fail at the end.

Part 3

For this problem we used a second predicate with an extra parameter (list of visited squares). For each square we get the next movement (calling move1) and then check if we have not visited that square yet. If the square has not been visited yet we now call the visit predicate from this new square. This is very similar to the *path* predicate implemented in part 1, but the difference is the base case. In part 1 the base case was when we reached the final node, in this part the base case is when N=0 (meaning no more movements are allowed)

Humam Altayeb

V00######