

# Welcome to the Test Problem for Robotics Engineer candidates!

You have 7 calendar days to solve the problem. If by the end of day 7 you have only a partial solution, please send it anyway with a respective comment. Please submit your Planner as a single zip file (with any readme/instructions inside) to [careers@instock.com](mailto:careers@instock.com). Don't hesitate to email us with any questions about the test problem.

-----  
Design and implement in code a Planner algorithm for a square chessboard of a given size (MxM) with certain cells marked red as restricted areas. 100x100 will be the maximum chessboard size. There are N kings on the board, each with a specific target cell to reach. The kings move sequentially, one by one, one step at a time (K1 step1, K2 step1.... K1 step2, K2 step2...), and must adhere to two main rules: they cannot move into restricted cells, and they cannot end a move adjacent to another king (including diagonally). A king may stay in place (empty move). The task is to compute a path for each king to its target, ensuring compliance with the movement rules and the sequential order of moves.

Planner's input would be given as 2 files:

File1 - king starting and target positions

x\_from, y\_from, x\_to, y\_to

...

File2 - chessboard size and restricted cells

M

x, y

...

x and y - integer coordinates, start from zero.

If plan (solution) does not exist, Planner should detect and output that plan does not exist.

If plan does exist, Planner should output a plan into a file **solution.txt** in a format

x\_from, y\_from, x\_to, y\_to

...

where each line is one step of one king, and king\_id would be deduced by a solution\_checker program from "from" coordinates.

Planner should finish planning in 10 minutes in any case. Meaning, Planner has to terminate within 10 minutes with at least some (maybe non-optimal) solution. We will be running tests on multi-core CPU, but we do not recommend spending time on concurrent implementations. Please do not implement anything for GPUs.

The Planner program should use not more than 1GB of RAM.

A reasonable simple visualization would be a plus but is not a requirement.

Your ideas on optimality of your algorithm would be a valuable add-on when discussing your solution on a call.

Implementations can be done in Go, C++ or python. You can use anything (libraries, packages...) available for public use (any open-source license works). But the more you understand what a library is doing the better.

Here is the data for two simple test inputs:

<https://drive.google.com/file/d/1hUi21VM0F6gz0PWI9NaJkECujijyfDm/view>