Jobrapido: Backend Engineer Test

The program should model a knight in his shiny armor moving on a board filled with obstacles. Following his king's commands, the knight should move on the board, following some simple rules and at the end of the excution the ouput should represent his position and direction.

The program must be executed in a Docker container (you have to provide the Dockerfile), outputing on standard output (console) a result JSON (definition of the output JSON below). We expect to been able to test your program running the following commands from the root of the project (feel free to provde a different command if you need for your project):

```
docker build -t knight_board:latest .
docker run \
-e BOARD_API=https://storage.googleapis.com/jobrapido-backend-test/board.json \
-e COMMANDS_API=https://storage.googleapis.com/jobrapido-backend-test/commands.json \
knight_board:latest
```

You can choose the language to use for the implementation among one of the following: Java/Kotlin, Golang.

Board API definition

The definition of the board must be retrieved from the following API: `https://storage.googleapis.com/jobrapido-backend-test/board.json`. The actual URL must be retrieved from `BOARD_API` environment variable.
The response of the API is:

```
{
    "width": 10, //Dimension of the board on x axis
    "height": 10, //Dimension of the board on y axis
    "obstacles": [ //List of points of the board where the knight cannot step on
        {
            "x": 1,
            "y": 1,
        },
        {
            "x": 1,
            "y": 2
        }
    ]
}
```

N.B.: consider 0,0 as the origin point for the board, positioned at the bottom left of it.

Command API definition

The list of commands to execute must be retrieved from the following API: `https://storage.googleapis.com/jobrapido-backend-test/commands.json`. The actual URL must be retrieved from `COMMANDS_API` environment variable.
The response of the API is:

```
{
    "commands": [
        "START 1,0,NORTH",
        "ROTATE SOUTH",
        "MOVE 3",
        "ROTATE EAST",
        "MOVE 5"
    ]
}
```

Commands definition

The king will tell his knight how to move on the board. Here a list of the command that the knight must be aware of:

- `START X,Y,DIR`: This is always the first command received by the knight. `X,Y` represent the starting position of the knight, `DIR` represent the direction is facing. `DIR` could be NORTH, SOUTH, EST, WEST.

- `ROTATE DIR`: This command made the knight turn itself facing the desired direction. `DIR` could be NORTH, SOUTH, EST, WEST. The kningt does not change his position when this command is executed.
- `MOVE N`: The knight will move by `N` steps forward (`N>=0`).

Expected output

The program should output on standard output (console) a json representing the outcome of the commands execution:

```
{
    "position": {
        "x": 1,
        "y": 1,
        "direction": "NORTH"
    },
    "status": "SUCCESS"
}
```

Status definition

- `SUCCESS`: when the whole list of commands has been executed without any error
- `INVALID_START_POSITION`: when the starting position is not valid
- `OUT_OF_THE_BOARD`: then the knight fell out of the board while executing the given commands
- `GENERIC_ERROR`: for all other possible error cases

N.B: When the status is not `SUCCESS` position information must be omitted.

Rules

- The starting position must be validated by the program: if the starting position is not valid the execution must end with the following status: `INVALID_START_POSITION` . The starting position is invalid when the coordinates overlapse with an obstacle or are out of the board.
- If the MOVE command bring the knight out of the board, you must terminate commands list execution with the following status: `OUT_OF_THE_BOARD`
- While executing the `MOVE` command, if the knight encounters an obstacle on his path, he will stop at the position before the obstacle, terminating the current `MOVE` command, proceding with the execution of the next command in the list, if any.
  E.g. If the command is `MOVE 5` and after 3 steps the knight encounter an obstacle, you have to ignore the last 2 requested steps.

Sample

The following image is a graphical representation of the board (described by the sample JSON), and the path travelled by the knight following the given sample commands. At the end you find the expected output for this sample. The red dot represent the starting point, and the knight is in the final position.

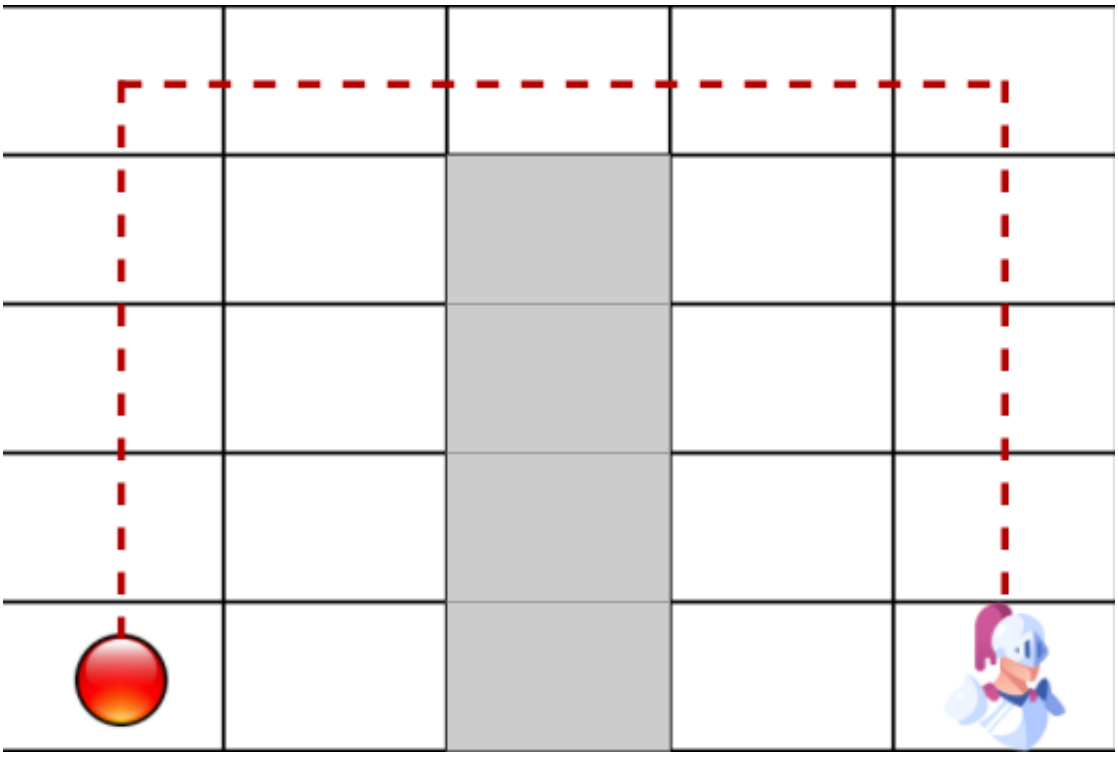

Figure 1: knight-path-sample

Board definition:

```
{
   "width":5,
   "height":5,
   "obstacles":[
      {
         "x":2,
         "y":0
      },
      {
         "x":2,
         "y":1
      },
      {
         "x":2,
         "y":2
      },
      {
         "x":2,
         "y":3
      }
   ]
}
```

Commands:

```
{
```

```
    "commands":[
       "START 0,0,NORTH",
       "MOVE 4",
       "ROTATE EAST",
       "MOVE 4",
       "ROTATE SOUTH",
       "MOVE 4"
    ]
}
```

Expected output:

```
{
    "position":{
       "x":4,
       "y":0,
       "direction":"SOUTH"
    },
    "status":"SUCCESS"
}
```

Guidelines

- Describe your thought process behind your choices and explain the reasoning behind your decisions
- While testing is important, full coverage is not expected. Demonstrate how you would approach testing by prioritizing the most critical aspects and selecting the most appropriate techniques.
- Do not forget the KISS Principle.