1 Tibber Platform Introduction

Tibber platform consists of swarm of micro services running as Docker containers. Primary development platforms are .NET Core and Node JS in conjunction with other platforms. Backed mostly using PostgreSQL as relational/document storage and Amazon S3 as blob storage.

2 Task

Create a new micro service that could fit into the Tibber Platform environment as described above. The service will simulate a robot moving in an office space and cleaning the places the robot visits. The path of the robot movement is described by start coordinates and move commands. After the cleaning has been done the robot reports the number of **unique** places cleaned, the service will store the result into the database and returns the created record in JSON format. The service listens to HTTP protocol on port 5000.

Request method: `POST`
Request path: `/tibber-developer-test/enter-path`
Input criteria:
$0 \leq$ number of `commmands` elements ≤ 10000
$-100\,000 \leq \mathtt{x} \leq 100\,000$
$-100\,000 \leq \mathtt{y} \leq 100\,000$
$\mathtt{direction} \in \{\mathtt{north}, \mathtt{east}, \mathtt{south}, \mathtt{west}\}$
$0 < \mathtt{steps} < 100\,000$

Request body example:

```
{
    "start": {
        "x": 10,
        "y": 22
    },
    "commmands": [
        {
            "direction": "east",
            "steps": 2
        },
        {
            "direction": "north",
            "steps": 1
        }
    ]
}
```

The resulting value will be stored in a table named `executions` together with timestamp of insertion, number of command elements and duration of the calculation in seconds.

Stored record example:

id	**timestamp**	**commmands**	**result**	**duration**
1234	2018-05-12 12:45:10.851596 +02:00	2	4	0.000123

3 Notes

- You can assume, for the sake of simplicity, that the office can be viewed as a grid where the robot moves only on the vertices.
- The robot cleans at every vertex it touches, not just where it stops.
- All input should be considered well formed and syntactically correct. There is no need, therefore, to implement elaborate input validation.
- The robot will never be sent outside the bounds of the office.
- Ensure that database connection is configurable using environment variable.
- Think about structure, readability, maintainability, performance, re-usability and test-ability of the code. Like the solution is going to be deployed into production environment. You should be proud of what you deliver.
- Use only open source dependencies if needed.
- Include `Dockerfile` and `docker-compose` configuration files in the solution.