MyQ Unattended Coding Test
Cleaning Robot

Revised date: 06/17/2016

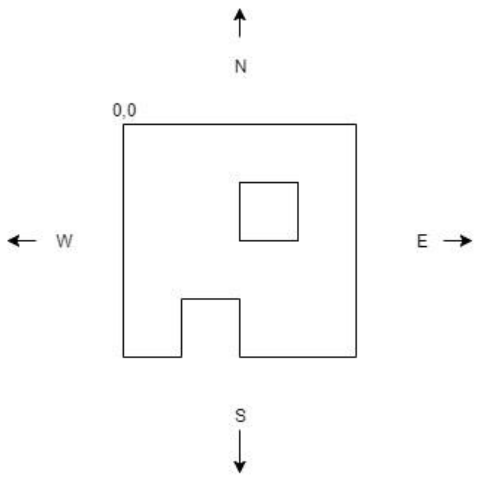
Description

MyQ has decided to launch a new automated cleaning robot to the market that can clean all surfaces in a house automatically without manual intervention.

To map the house, the robot will receive information about the room as a set of cells. Each cell represents:

- A cleanable space of 1 by 1 that can be occupied and cleaned (S).
- A column of 1 by 1 which can't be occupied or cleaned (C)
- A wall (represented by an empty cell or by being outside the matrix) (null)

This map is provided as a matrix of m by n in which each element of the matrix is one of those items or else it is empty (null). For example, the 4x4 map (top left is always 0,0):



```
[
[S, S, S, S]
[S, S, C, S]
[S, S, S, S]
[S, null, S, S]
]
```

The robot also receives a set of basic instructions, each of which drain the battery of the robot by a certain amount and that it must follow.

- Turn Left (TL). Instructs the robot to turn 90 degrees to the left. Consumes 1 unit of battery.
- Turn Right (TR). Instructs the robot to turn 90 degrees to the right. Consumes 1 unit of battery.
- Advance (A). Instructs the robot to advance one cell forward into the next cell. Consumes 2 unit of battery.
- Back (B). Instructs the robot to move back one cell without changing direction. Consumes 3 units of battery.
- Clean (C). Instructs the robot to clean the current cell. Consumes 5 units of battery.

The Back (B) command is not permitted in the received command set; it is used only to implement a back off algorithm when the robot hits an obstacle. A sequence of valid operations will look like:

- C, TR, A, C, A, C, A, C, TL, A, C, A, A, A, C, TL, C

The robot will process each operation and must obey it unless it hits an obstacle (a column or a wall) or runs out of battery. If the command (which can be only Advance) would result in the robot entering an obstacle cell, the robot will ignore the order and instead initiate the following algorithm (back off strategy):

1. Turn right, then advance. (TR, A)
2. If that also hits an obstacle: Turn Left, Back, Turn Right, Advance (TL, B, TR, A)
3. If that also hits an obstacle: Turn Left, Turn Left, Advance (TL, TL, A)
4. If that also hits and obstacle: Turn Right, Back, Turn Right, Advance (TR, B, TR, A)
5. 5. If that also hits and obstacle: Turn Left, Turn Left, Advance (TL, TL, A)
6. 6. If an obstacle is hit again the robot will stop and return.

The robot will execute each command in order until no more commands are left, the battery is completely drained or all the back off strategies fail. A command can only be executed if there's enough battery to complete it. If, for example, the robot has 4 units of battery and a "clean" command is received, the robot will stop at that point and finish the program with 4 units left of battery.

To provide the information to the robot, a json file is provided with the following format:

```
{
  "map": [
    ["S",  "S",  "S",  "S"],
    ["S",  "S",  "C",  "S"],
    ["S",  "S",  "S",  "S"],
    ["S",  "null",  "S",  "S"]
  ],
  "start": {"X": 3,  "Y": 0,  "facing": "N"},
  "commands": [ "TL","A","C","A","C","TR","A","C"],
  "battery": 80
}
```

In which "map" contains the map; "start" contains the starting point of the robot as X, Y with X being the column and Y being the row, for example, the column obstacle in the example is on position (2,1) and the direction it is facing which can be North (N), East (E), South (S) or West (W); "commands" contains the ordered list of commands to execute, "battery" the initial battery level.

Upon execution, the robot must produce a result json which describes the results of the cleaning containing:

- All cells visited.
- All cells cleaned.
- Final position of the robot.

- Final battery left.

```
{
  “visited” : [ {“X”: 1, “Y”:0}, {“X”: 2, “Y” : 0}, {“X”:3, “Y”: 0}],
  “cleaned”: [{“X”: 1, “Y”:0}, {“X”: 2, “Y” : 1}],
  “final”: { “X” : 2, “Y” : 0, “facing”: “E”},
  “battery” : 54
}
```

Task

- Create a command line program that will receive a json file as an input with the parameter specified and will run the robot simulation and produce the output as specified: ○ Visited will contain all the cells visited by the robot ○ Cleaned will contain all the cells cleaned by the robot ○ Final will contain the final position and facing direction of the robot.
  ○ Battery will contain the final battery left on the robot.
- The program must be able to be invoked as:

```
cleaning_robot source.json result.json
```

- The program must take the first parameter (source.json on the example) as the input and produce the results in the second parameter (result.json).

Deliverables

- Your solution may be delivered on any language of your choice.
- Please include instructions on how to build and run your code.
- Don’t forget to include any tests or quality assurance measures you see appropriate.

Bonus points

- Bonus point for creating a full REST microservice in addition to the console app - Bonus point for usage of source control.
- Bonus point for usage of containerization.

Guidance.

- The coding tests includes good design patterns, reusability and extensibility.
- We expect candidates to create code that is production quality – the solution should be correct, reliable, maintainable, reusable, portable and efficient.
- Include documentation to justify the approach taken.
- SOLID principles will be valued.
- In the provided ZIP file, together with the description of this problem, there're two json example for both a sample input and the expected produced output.