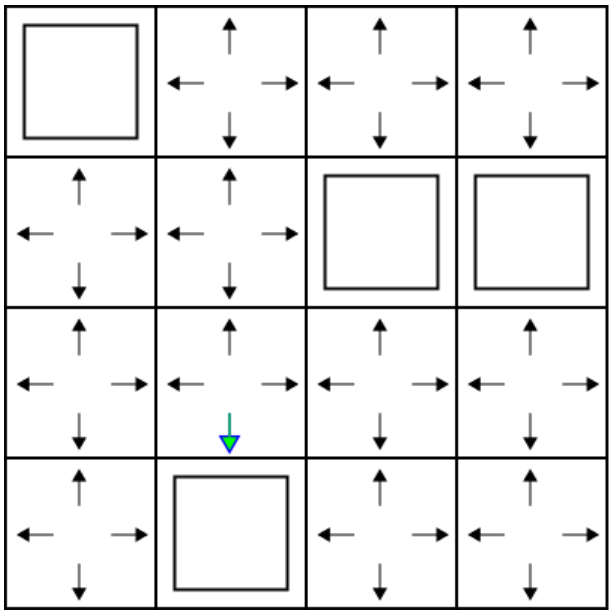
Lab 5: Orienteering

Objective

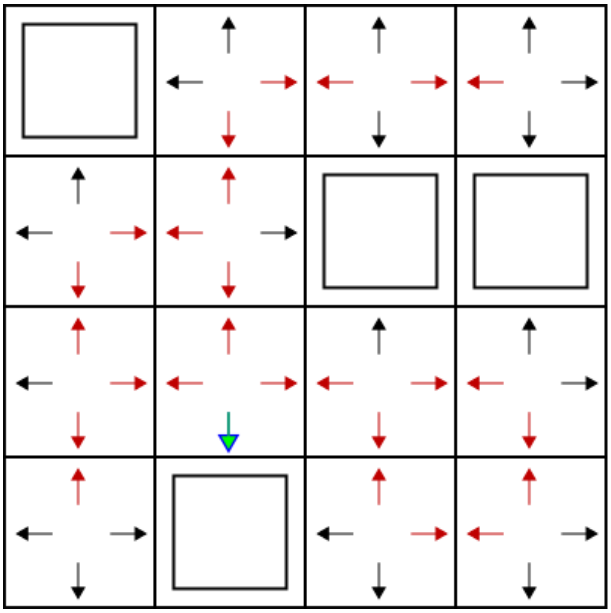
The objective of this lab is to implement an algorithm that enables the robot to determine its location and orientation in an enclosed (i.e. surrounded on all four sides by walls) 4-by-4 grid tile area, using the known position of obstacles as 'landmarks' with which to orient itself. The robot will start facing one of the four cardinal directions (north, east, south, or west, that is to say, in the positive-y, positive-x, negative-y, or negative-x directions, respectively) at the centre of an unobstructed tile. Once the robot has determined its position, it is to travel to the tile in the northeast corner of the area and face north (i.e. if the southwest gridline intersection is considered to be the origin, it is to travel to (75, 75) and face in the positive-y direction). Note that *both the starting tile and starting orientation of the robot are picked arbitrarily by the TA at the start of* ***each*** *demo.*

Algorithm

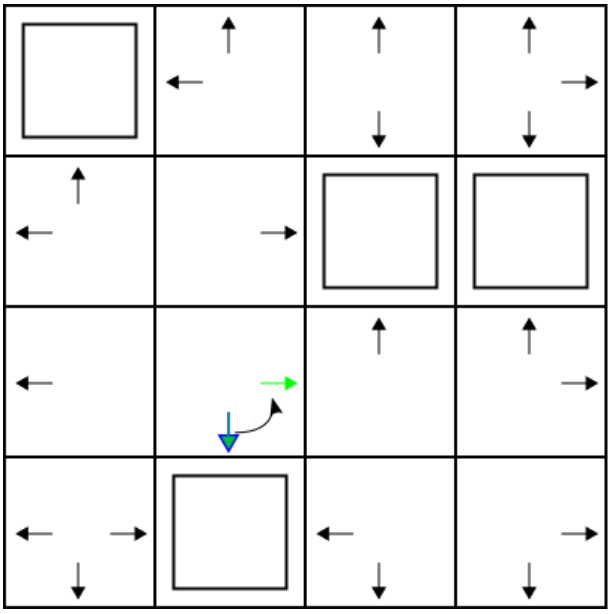
There are many possible approaches to this problem, but one is to exclude impossible starting locations by consideration of the *relative* path of the robot from its (arbitrary) starting location, as well as consideration of the presence (or absence) of obstacles/walls in adjacent tiles. For the purpose of this lab, the field layout shown below will be used, however the obstacles are not to-scale.



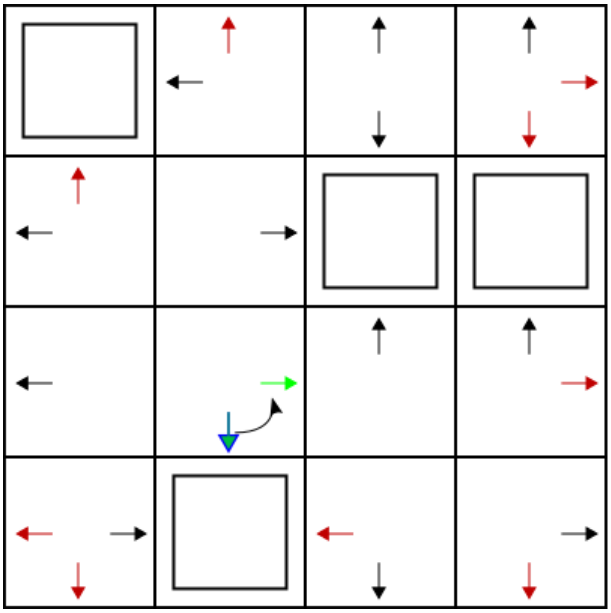
In the above image, the robot's starting position is assumed be the blue arrow (this arrow will remain present on subsequent images for reference), and the green arrow represents its current location. The black arrows indicate all the starting positions that the robot has yet to rule out.



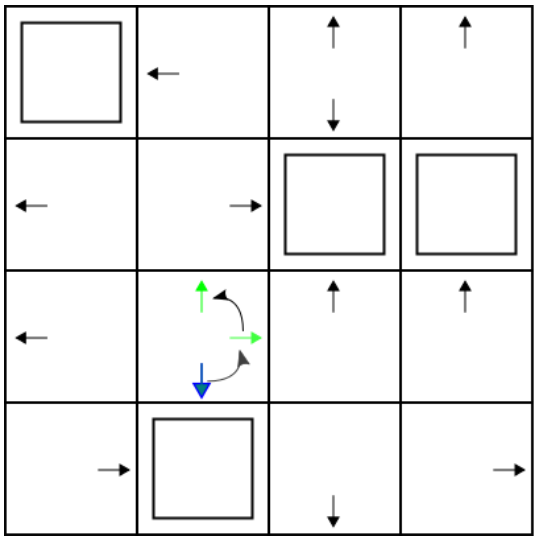
By observing (using the ultrasonic sensor) that it is facing an obstacle or wall, the robot can immediately exclude the red arrows as possible starting locations.



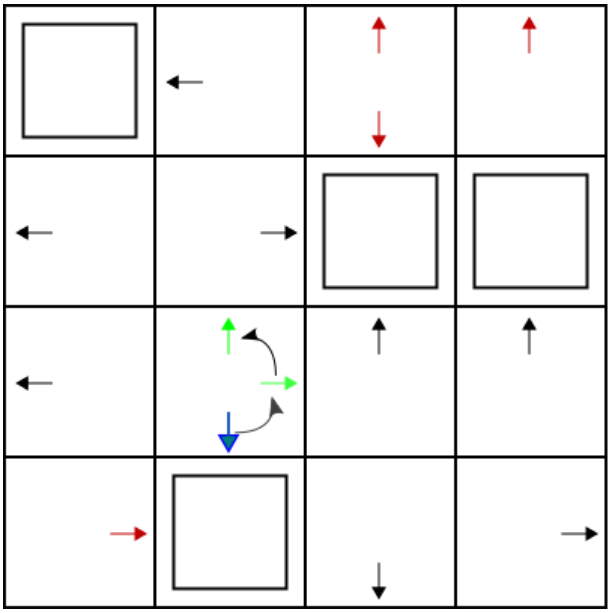
In order to continue ruling out possible starting positions, the robot must reorient or move. In this case, the robot rotates 90 degrees counter-clockwise. Note that in this example, the robot's choice of movement is arbitrary. *(Hint: A more clever algorithm would consider which motions would rule out the greatest number of the remaining possible starting locations.)*



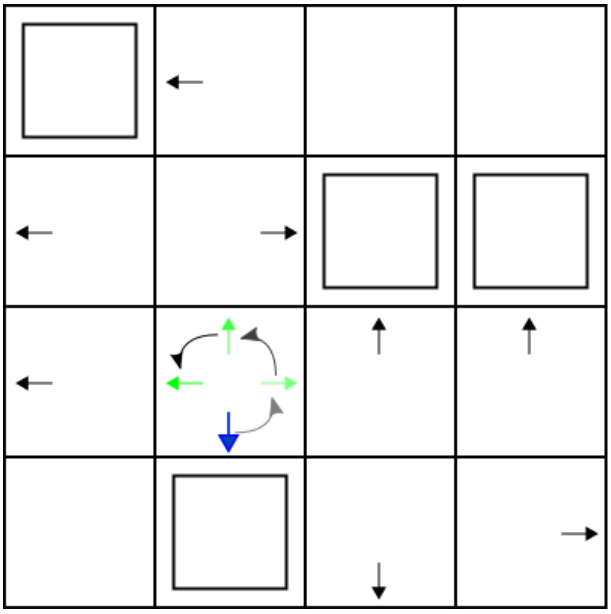
It then observes no obstacle directly in front of it, and thus can rule out all starting positions with an obstacle or wall 90 degrees to their left.



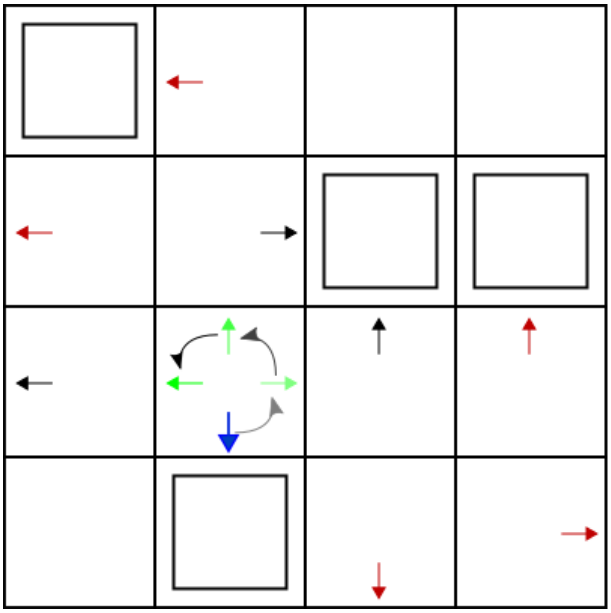
Again, it rotates 90 degrees counter-clockwise (another arbitrary choice for the purpose of this example – that is, it could alternatively have moved forward one tile).



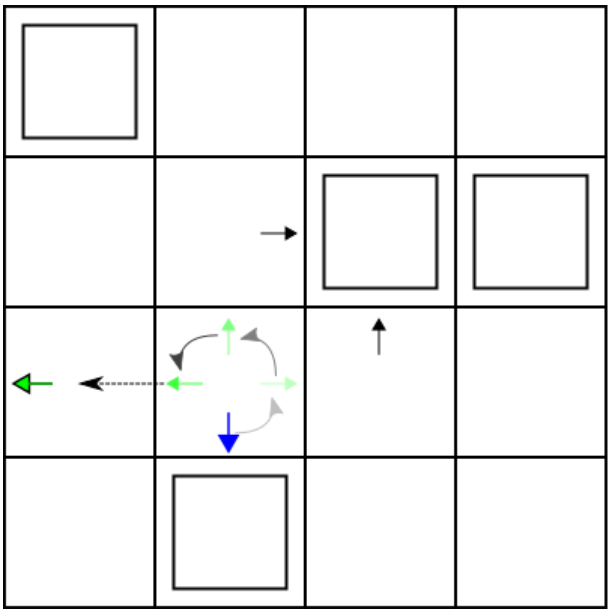
By observing that directly behind its starting position is a vacant tile, it can exclude starting positions with obstacles or walls behind them.



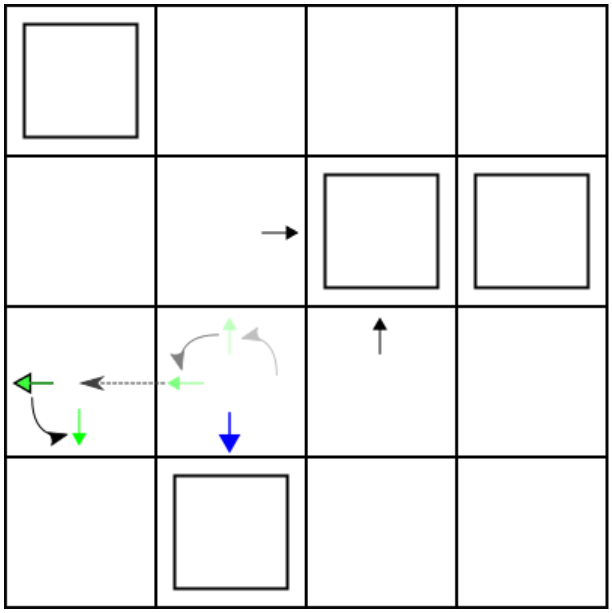
It then rotates 90 degrees counter-clockwise so that it is now facing to the right of its initial starting position (note the position of the darkest green arrow relative to the blue arrow).



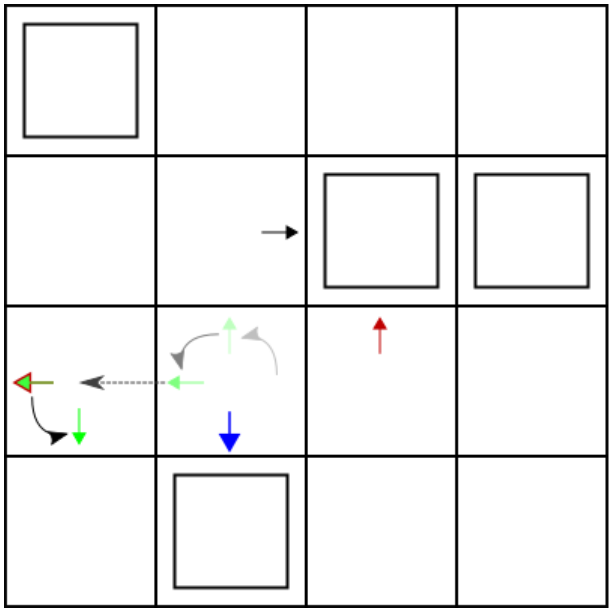
After this motion, it is able to rule out starting positions with an obstacle or wall 90 degrees immediately to their right. Note that the four remaining starting positions (i.e. the three black arrows and the blue arrow) are those that face an obstacle, but have no obstacles to their left, their right, or behind them.



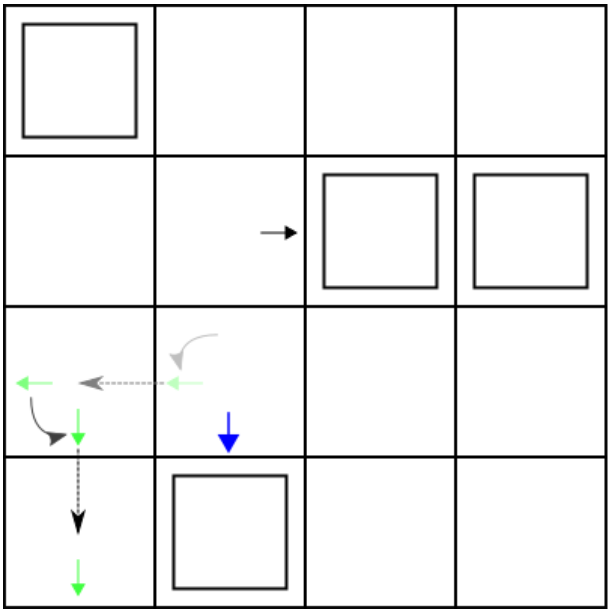
At this point, rotating an additional 90 degrees would return it to its initial position, and the subsequent observation would yield no new data. Instead, it moves forward one tile. Note that the green arrow on the left (the robot's position after the motion forward) has a black outline, as this is one of the remaining valid starting positions. In this position, the robot observes an obstacle – relative to its starting position, this obstacle is two tiles to its right. However, all four of the remaining possible starting positions *also* have an obstacle or wall two tiles to their right, so none of them can be ruled out in this step.



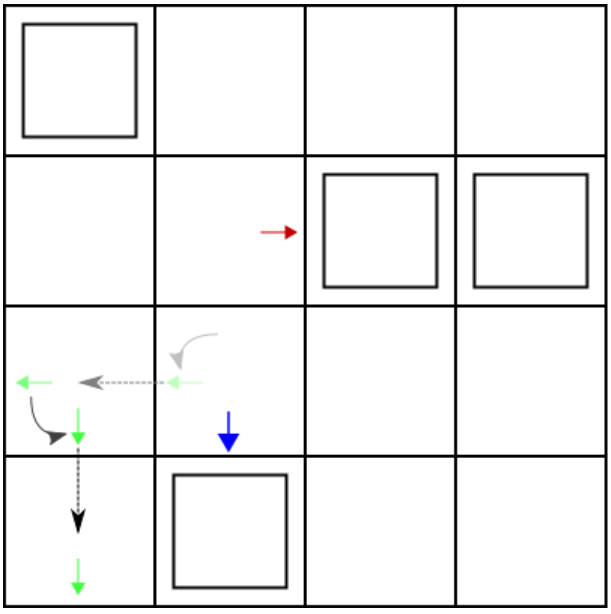
The robot then rotates 90 degrees counter-clockwise (again, it is stressed that this particular choice of path is provided *solely* as an example, and has no rhyme nor reason beyond demonstrating the operation of the algorithm).



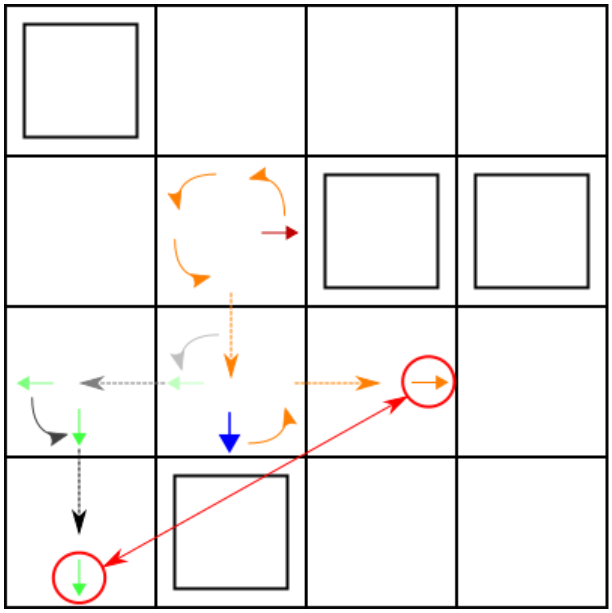
Here, the robot observes no obstacle, and can exclude two of the remaining four starting positions (that located on the far left, under the green arrow representing the robot's previous position, and that pointing northward in red). Note that the tile it is observing is, relative to its starting position, forward and to the right of its starting position – the blue and black arrows have this tile vacant, whereas the two red arrows do not.



The robot then moves forward one tile, as rotating 90 degrees counter-clockwise would cause it to face a tile it had previously occupied (and thus provide no new information, lengthening unnecessarily this example).



Finally, it observes an obstacle or wall directly in front of it, two tiles forward and one tile to the right of its initial starting position (that is, 'forward' and 'right' relative to its initial starting orientation). Note that the same does not hold for the red arrow – the same tile relative to it, notably that on the far east side of the field, one north of the bottom row, is vacant. Thus, it can remove the red arrow as a potential starting position, and as the blue arrow is the only position remaining, the robot has successfully determined its starting position and orientation and, with the knowledge of the relative motions it made during the orienteering process, its current position and orientation on the field.



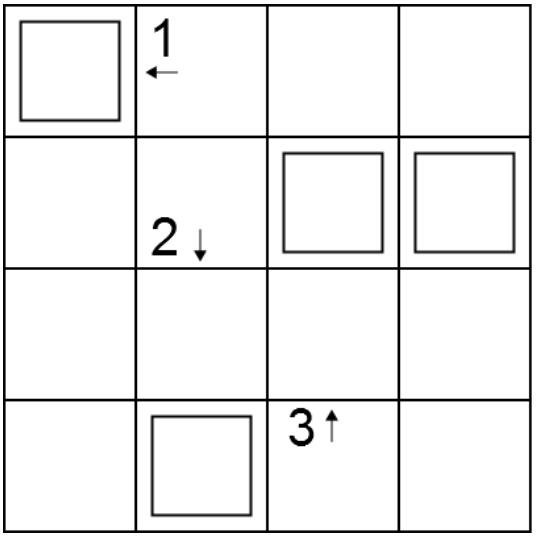
If it is unclear as to *why* the red arrow is no longer a valid starting position after the observation made by the robot in the last step, consider the above graphic. In orange are the same motions (in the same order) performed by the robot, rotated and translated to start at the red arrow. The last observation of the robot was that the tile directly in front of it was obstructed (in fact, it is a wall), however, had it started at the red arrow, it would have observed a vacant tile directly in front of it – thus the red arrow cannot represent a possible valid starting location.

This is the same logic used at each step of this algorithm; for brevity, only the last exclusion is explained in detail, but implementers of this or similar algorithms are encouraged to work out the logic for some of the other exclusions in previous steps until they are satisfied with their understanding of the algorithm.

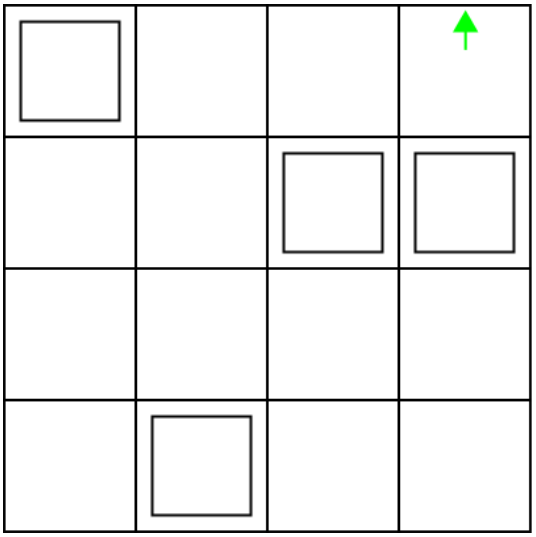
Method

1. Implement an algorithm similar to the one outlined above to determine the position and orientation of the robot on the field. Your algorithm should have two variants:
    a. Deterministic: The robot performs an observation of what is directly in front of it, excludes starting positions that do not correspond to the observation, and then performs one of two motions: if it has an obstacle or wall directly in front of it, it rotates 90 degrees counter-clockwise; if it does not, it moves forward one tile. The robot repeats this observation/motion process until all but one starting position is excluded.
    b. Stochastic: The robot performs an observation of what is directly in front of it, excludes starting positions that do not correspond to the observation, and then performs one of two motions: if it has an obstacle or wall directly in front of it, it rotates 90 degrees counter-clockwise; if it does not, it chooses *randomly* to *either* rotate 90 degrees counter-clockwise *or* move forward one tile (with equal odds for each). The robot repeats this observation/motion process until all but one starting position is excluded.

2. Add a counter to orienteering algorithm to count the number of observations it makes (this should be one larger than the number of motions when the algorithm terminates, as redundant observations are to be included). Once the orienteering algorithm terminates, display this number on the screen.
3. For each of the three starting positions indicated below, run the deterministic variant of your algorithm once, and the stochastic variant ten (10) times (i.e. a total of 33 runs). Record the number of observations made by the deterministic algorithm at each starting point, and the number of observations made by the stochastic algorithm at each starting point, for each run. You should provide a 11-by-3 table (i.e. 11 rows and 3 columns, not including a header row and header column) indicating the number of observations made in each of the 11 runs (10 stochastic, 1 deterministic) for each of the three starting points.



4. Modify your code to make your robot move to the position and orientation seen in green below after it has completed its orienteering algorithm.



5. Demonstrate the deterministic variant of your algorithm to a TA. The TA will place your robot at the centre of *any* unobstructed grid tile (*not necessarily* one of the three you used for step 3) facing one of the four cardinal directions. For a successful demonstration, your robot should:
    a. Follow the steps of the deterministic algorithm as specified in 1a
    b. After completing its orienteering, move to the position and orientation specified in 4
    c. Finally, display the position the robot thought it started in on the screen (e.g. '-15, -15, E' would be the southwest corner, facing east as a starting location)
    d. Complete parts (a), (b) and (c) in less than 4 minutes

Data

- One table with columns labelled "Position 1," "Position 2," and "Position 3," and 11 rows labelled "Stochastic 1," "Stochastic 2," …, "Stochastic 10," and "Deterministic" whose entries consist of the number of observations (i.e. obstacle/wall or no obstacle/no wall) made by the robot for the corresponding run and starting position.
- The mean number of observations performed by the stochastic algorithm for each of the three starting positions.

Data Analysis

1. Did the stochastic algorithm, on average, perform better or worse that the deterministic algorithm? Explain why or why not.
2. In the worst case, will either algorithm terminate (for the given field layout)? Explain why or why not.
3. Do all obstacle layouts allow for orienteering (*hint: consider an empty field*)? If so, explain why. If not, explain what characteristic(s) a field layout must have in order to ensure that a robot can uniquely determine its starting position, and provide an example of a field layout that *does not* allow a robot to uniquely determine its starting position.

Further Improvements

1. The ultrasonic sensor is capable of detecting objects at distances greater than 30 cm. Explain how this fact could be used to improve the number of possible starting positions at each step, and comment on the possible shortcomings of this approach.
2. The stochastic algorithm you designed, when given the option, rotated 50% of the time, and moved forward 50% of the time, however these odds are chosen arbitrarily. The algorithm can be modified to behave less randomly and more like the deterministic algorithm while still, fundamentally, remaining stochastic. Explain how this can be achieved, and give your opinion (supported with arguments) as to how this would (or would not) be an improvement.
3. Neither of the algorithms used in this lab choose the motions of the robot based on which starting positions remain to be excluded. Propose a means by which the robot could *choose* the next motion (and consequently, the next observation) it should make so as to efficiently rule out possible starting positions. You do not need to provide code (a description, in English or French

of the algorithm's operation will suffice), but your algorithm should be scalable (i.e. it should work on the 12-by-12 field that will be used in the final project).

To Sumbit

- One document in .pdf format containing the lab report (if scanned work is included, you may either attach the scan and zip the file, or put the scans into the document)
- NOTE: Lab reports should be kept concise, and to the point. No cover page is needed when submitting the weekly lab reports.