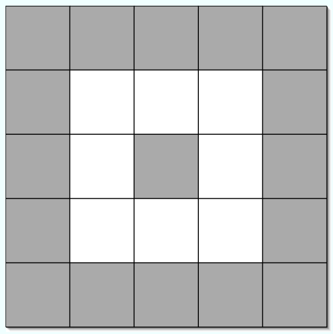
# Introduction to Robotics - CMSC 479/679
# Homework #3
**Due Wednesday, April $25^{th}$ at the start of class**

The purpose of this homework assignment is to give you experience with Markov localization by implementing the algorithm for a very simple robot that lives in a grid world. Below are specifications for both the robot and its world.

**World:** Assume the robot lives in a rectangular world that has been discretized via a fixed cell decomposition. The robot's world $W$ can be represented as a 2-dimensional array of integers, with $W(i, j)$ indicating whether the cell in row $i$ and column $j$ contains an obstacle or is open. For example, you might set it up such that if $W(i, j) = 0$ then the corresponding cell is open, and if $W(i, j) = 1$ then the corresponding cell contains an obstacle. All of the cells on the border of the world should contain an obstacle so that the robot cannot wander off the edge. Consider the very simple world below, in which dark cells are obstacles and light cells are open.



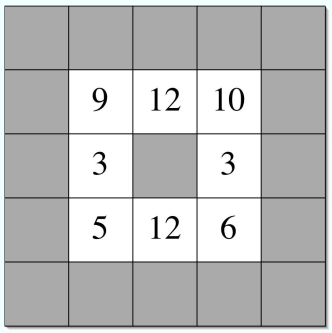
This world might look like the following represented as a 2D array.

```
1 1 1 1 1
1 0 0 0 1
1 0 1 0 1
1 0 0 0 1
1 1 1 1 1
```

The worlds that you create should be larger and more complex than this example, but shouldn't be so large as to require excessive computation or memory.

**Robot:** The robot can occupy a single open cell in the world at any given time. It can take one of four actions – N, S, E, W – which correspond to moving once cell to the north, south, east, or west. If the robot attempts to move into a cell occupied by an obstacle, its position remains unchanged. Otherwise, with probability $p$ it moves one cell in the desired direction, and with probability $1 - p$ it moves two cells in the desired direction (unless moving two cells would put it into a cell occupied by an obstacle, in which case it will only move one cell). Note that when $p = 1$, the robot's odometry is perfect.

The robot has four sensors that always point in one of the four compass directions. (Don't ask me how, they just do.) Call those sensors N, S, E, and W. That is, no matter where the robot is, sensor N always returns information about the cell adjacent to the robot to the north. If the cell adjacent to the robot in the sensor's direction contains an obstacle it returns a 1, otherwise it returns a 0. However, with probability $q$ the value returned is toggled to simulate noise. Note that when $q = 0$ the robot's sensors are perfect. It is easiest to think of the sensor values as comprising a four-bit number that represents the robot's sensor readings as an integer between 0 and 15. For example, below is the sample world with the open cells containing the integer sensed by the robot in that cell.



The value in the upper left cell is 9 because in that cell $N = 1, S = 0, E = 0, W = 1$, so $NSEW = 1001 = 9$. Likewise, the value in the lower right cell is 6 because in that cell $N = 0, S = 1, E = 1, W = 0$, so $NSEW = 0110 = 6$.

Your tasks are as follows:

- Implement a small world for your robot to inhabit with a few rooms that contain a few objects. Note that rooms can be defined by creating walls that consist of obstacle cells, and objects can be defined by creating blocks of adjacent obstacle cells.
- Implement a robot as described above that can be "driven" around in the world, with noisy sensors and effectors.
- Implement the Markov localization algorithm as defined in the book, using the map of the world you created along with the models of the robot's sensor and effector noise described above. The model of effector noise will be used to update location probabilities in the ACT part of the algorithm, and the model of sensor noise will be used to update location probabilities in the SEE part of the algorithm. Be sure to assign uniform probability to all cells when the robot is first "switched on" in the world.
- Come up with a scheme for driving your robot around. The simplest possible scheme is a random walk, in which you choose one of the four actions with equal probability on each time step.
- Experiment with different values of $p$ and $q$, and, if you want to, different worlds to see how the robot's belief state evolves over time.

You should turn in the following:

- All source code, in any language you choose.
- Picture(s) of the world(s) you used.
- Some graphical depiction of the belief state as it evolves over time for at least one run of the system. For example, a plot of each cell where grayscale intensity is correlated with probability would be great. Specify the action selection strategy and the values of $p$ and $q$.
- Write a brief report explaining the experiments you ran, describing in qualitative terms the impact of varying $p$ and $q$, and anything else of interest you observed.