Assignment 1 - AI 2016 - Minor Robotica

- **It will be graded.**
- **Execute in groups of two persons.**
- **Create one PDF per group with the answers to the questions on paper and include a short but clear explanation of what you programmed.**
- **You will only alter the sampleAgent.py class**
- **Combine the PDF + your altered sampleAgent.py code file in one ZIP file with the name:**
  AI2015 VAC lastname1-studentnr1 lastname2-studentnr2.zip
  so e.g.:
  AI2015 VAC deKok-500223133 Stolk-500131323.zip
- **Upload the zip-file to the DLWO-Dropbox of this course.**
  Other naming conventions will not be graded.

The code provided originates from Christopher H. Brooks, professor at the University of San Francisco, Computer Science AI Course 662, thanks go out to the original author.

In the book you've read about the simple AI form of the vacuum cleaner robot. The given implementation is a simplified form of the Modern Approach book's python code. This code implements a random agent and a table based agent to vacuum clean the rooms.

For this homework, you'll extend some existing code to build a set of agents that can successfully operate in the Vacuum Environment.

First download **Python 2.7** from http://www.python.org/download or install an entire suite (recommended for future projects) http://python-xy.github.io/ and install it to your system.

Next, examine the given code: check out the available functions, what they do and when they do it. As (almost) always: it will benefit you in the long run if you take first a proper amount of time to get a feeling of the structure/functions of the (given) software you are dealing with.

The provided software contains of the following files:

- environment.py provides an abstract Environment class, as well as a NbyMVacuumEnvironment class.

- sampleAgent.py provides an abstract Agent class, as well as RandomAgent and TableDrivenAgent classes, as starting points for you.

Start by running the existing code on the NbyMVacuumEnvironment, just to get comfortable with how it works. Here's a sample session from the command prompt (cmd):

```
python
>>> import environment
>>> import sampleAgent
>>> v = environment.NbyMVacuumEnvironment()
>>> ag = sampleAgent.RandomAgent(sampleAgent.agent_actions)
>>> v.add_agent(ag)

>>> v.run(10)
Initial room config:  {(1, 2): 'Dirty', (1, 1): 'Clean', (0, 2):
'Dirty', (1, 0): 'Dirty', (0, 0): 'Dirty', (0, 1): 'Dirty', (2, 0):
'Clean', (2, 1): 'Dirty', (2, 2): 'Dirty'}
Percept: ((1, 0), 'Dirty') Action: Left
Location: (1, 0) Action: Left
Percept: ((0, 0), 'Dirty') Action: Left
Location: (0, 0) Action: Left
Percept: ((0, 0), 'Dirty') Action: Suck
Location: (0, 0) Action: Suck
Percept: ((0, 0), 'Clean') Action: Suck
Location: (0, 0) Action: Suck
Percept: ((0, 0), 'Clean') Action: Left
Location: (0, 0) Action: Left
Percept: ((0, 0), 'Clean') Action: Suck
Location: (0, 0) Action: Suck
Percept: ((0, 0), 'Clean') Action: Suck
Location: (0, 0) Action: Suck
Percept: ((0, 0), 'Clean') Action: Suck
Location: (0, 0) Action: Suck
Percept: ((0, 0), 'Clean') Action: Suck
Location: (0, 0) Action: Suck
Percept: ((0, 0), 'Clean') Action: Left
Location: (0, 0) Action: Left
Final room config:  {(1, 2): 'Dirty', (1, 1): 'Clean', (0, 2): 'Dirty',
(1, 0): 'Dirty', (0, 0): 'Clean', (0, 1): 'Dirty', (2, 0): 'Clean', (2,
1): 'Dirty', (2, 2): 'Dirty'}
>>>
```

As you can see, the random agent isn't very bright (it just randomly wanders around), but it should give you a starting point. In particular, notice that the agent starts in a random location; your program cannot assume a particular starting point.

Next, run the TableDrivenAgent in the 3x3 environment. Unfortunately, the TableDrivenAgent is not able to guarantee that all of the rooms will be cleaned.

(1) Concisely explain why it is that the TableDrivenAgent is unable to accomplish this task.

Next, create a subclass of Agent called ModelBasedAgent. The ModelBasedAgent is able to keep a 'model' (or state, or memory) of the environment. (You may choose to implement this however you wish - it does not need to be very elaborate). You will need to implement two methods (you may find that you want helper methods also):

```
__init__()
program()
```

(2) Is your ModelBasedAgent guaranteed to eventually clean all rooms in a 3x3 grid environment? If so, show why. If not, show why not. Note: please do not solve the problem by modifying the TableDrivenAgent to choose random actions. This will eventually visit all the rooms, but doesn't really get at the point of the assignment.

(3) Is your ModelBasedAgent guaranteed to be optimal? In other words, does it clean all rooms in the minimum possible number of moves? If so, prove that this is the case. If not, explain what limitations of your agent prevent it from solving the problem optimally. (It's not required that your agent be optimal - just that you indicate whether it is or not.)

(4) Now let's make the problem more challenging: build a ModelBasedAgent that can solve the NxMVacuumEnvironment for other values of N and M. Notice that the values of N and M are NOT available to the agent; the only way it can figure out how large the grid is, is through exploration. In other words, you cannot modify the NByMVacuumEnvironment; just the agent.

Hint: how can an agent figure out whether there is another room to the south/east, given that its percepts tell it only the coordinates of the current room and whether it is clean or dirty? You might want to modify your model somewhat.