Homework #1: Chapters 1, 2, 3

1. [15 points] These answers are just examples, and there is actually quite a lot of latitude in terms of what could be correct. The important criteria are that performance measures can be quantified, that the environment contains the most important objects and agents that the agent will interact with, that the actuators are actually mechanisms that can change the environment, and that the sensors are mechanisms for receiving information about the environment (e.g., a microphone is a sensor not an actuator). [5 pts each environment, 1 pt each cell.]

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Performance Measure (Quant)</th>
<th>Environment (Env/Agent=Ext/Int)</th>
<th>Actuators</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>% of games won</td>
<td>deck, tableau piles, foundation piles</td>
<td>move card/pile api, turn deck cards api</td>
<td>receive state of both piles and deck</td>
</tr>
<tr>
<td>(b)</td>
<td>Life found, area covered</td>
<td>Mars: rocks, sand, hills, ditches</td>
<td>Steering, accelerator, brake, camera, life detection instruments</td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td>high # of correct fraud detections, low # of false detections</td>
<td>credit transactions</td>
<td>alert system (to notify of fraud)</td>
<td>receive transactions</td>
</tr>
</tbody>
</table>
2. **[15 points]** The important thing is that any answer given is consistent with the PEAS description and that explanation is plausible and reflects an understanding of the property. **[5 pts. each environment, 1 pt each cell]**

<table>
<thead>
<tr>
<th>Dimension</th>
<th>(a) Solitaire</th>
<th>(b) Mars rover</th>
<th>c) Credit fraud</th>
</tr>
</thead>
<tbody>
<tr>
<td>observable</td>
<td>partially (the agent cannot see the cards in the deck nor those that are hidden in the tableau pile)</td>
<td>partially (e.g., can’t see around rock)</td>
<td>fully (transactions are accurate/complete)</td>
</tr>
<tr>
<td>deterministic</td>
<td>deterministic (actions have expected effects, the identity of every hidden card and order of deck is determined before player starts to play)</td>
<td>stochastic (motors are imperfect, weather conditions)</td>
<td>stochastic (no control over next transactions)</td>
</tr>
<tr>
<td>episodic</td>
<td>sequential (each move determines what moves are subsequently available)</td>
<td>sequential (long-term consequences of short-term actions)</td>
<td>episodic (receives all transactions at once and decides)</td>
</tr>
<tr>
<td>static</td>
<td>static (the environment doesn’t change no matter how long the player takes )</td>
<td>dynamic (e.g., dust storms, weather conditions)</td>
<td>dynamic/semidynamic (detecting fraud too late isn’t as useful)</td>
</tr>
<tr>
<td>discrete</td>
<td>discrete (there are only 52 cards, a finite number of possible states)</td>
<td>continuous (camera images, steering angles range in values)</td>
<td>discrete (transactions have finite date, payees)</td>
</tr>
<tr>
<td># of agents</td>
<td>single (that’s the very definition of solitaire)</td>
<td>single (we assume there are no Martians or other rovers)</td>
<td>single (the defrauders are treated as part of the environment)</td>
</tr>
</tbody>
</table>
3. [10 points] Answers don’t have to match these exactly. Give plenty of partial credit for a reasonable attempt.

Some advantages of the table-driven agent (TDA) over the simple reflex agent (SRA): (5 pts.) TDA considers percept history, thus the next action can in a sense be more informed. As a result, the TDA can work in situations in which the SRA would fail (e.g., if an action returns one to the same state, the TDA may be able to “break out of the loop”).

Some advantages of the SRA over the TDA: (5 pts.) SRAs are simple and non-laborious to code. The TDA requires the programmer to explicitly encode for every possible history sequence. Since percepts are converted to “conditions”, the number of possibilities that the programmer must explicitly account for is also reduced. Thus it is easier to change the SRA’s behavior since there are far fewer state-action pairs than in the TDA. The SRA can also (practically) work in environments with a large state-space where the TDA is impractical/impossible to implement.

4. [20 pts.] Note clarification of homework on website: “the SUCCESSOR-FN is not defined. Assume SUCCESSOR-FN(s) returns a set of <a,s’> pairs, where a is an action that can be performed in s and s’ is a state that results if a is performed in s. Also, LEGAL-ACTIONS is just the ACTIONS function as described on p. 67.“

Both pseudo-code and mathematical definitions are given below:

Pseudo Code:

```plaintext
function SUCCESSOR-FN(s) returns set of <a, s’> pairs
    pairs, initially empty
    for each a in ACTIONS(s)  // LEGAL-ACTIONS is okay
        s’ = RESULT(a, s)
        add <a, s’> to pairs
    return pairs

function RESULT(a, s) returns state
    sucs = SUCCESSOR-FN(s)
    find <a_i^R, s_i^F> in sucs where a_i^R = a
    return s_i^F

function ACTIONS(s) returns set of actions  // LEGAL-ACTIONS is okay
    actions, initially empty
    sucs = SUCCESSOR-FN(s)
    for each <a_i, s_i> in sucs
        add a_i to actions
    return actions
```

Or, mathematically:

```
SUCCESSOR-FN(s) = {<a, s’> | a ∈ ACTIONS(s) and RESULT(a,s) = s’}
RESULT(a, s) = { s_i^F | <a_i^R, s_i^F> ∈ SUCCESSOR-FN(s) and a_i^R = a}
ACTIONS(s) = {a_i | <a_i, s_i> ∈ SUCCESSOR-FN(s)}
```
5. [20 points] Again this is just one possible solution, any solution that has an equivalent level of detail should do

Initial State: (5 points)
Use a factored representation. Some configuration consisting of a set of areas $N$, an adjacency function $\text{adjacent}(x) = \{y \mid \text{such that } y \text{ is adjacent to } x\}$ and a coloring function $\text{color} : N \rightarrow \{1,2,3,4,\text{uncolored}\}$.

Actions Function: (5 points)
for each state $s$:
- for each $x \in N$ such that $\text{color}(x) = \text{uncolored}$
  - for each $c \in \{1,2,3,4\}$ such that there does not exist a $y \in \text{adjacent}(x)$ where $\text{color}(y) = c$
    - [set $x$’s color to $c$] $\in \text{ACTIONS}(s)$

Result Function: (5 points)
for each state $s$,
- for each $a \in \text{ACTIONS}(s)$ where $a = [\text{set } x \text{’s color to } c]$
  - $\{a, s'\} \in \text{RESULT}(s)$ where $s'$ is $s$ with color() changed so that $\text{color}(x) = c$

Goal Test: (3 points)
true if there is no $x \in N$ such that $\text{color}(x) = \text{uncolored}$.

Path Cost Function: (2 points)
All that matters for this problem is that a solution is found. No solution is considered any better than any other. Thus path cost could be 0 or 1. A path cost calculated on 1 per step is also acceptable, since all solutions require the same number of steps (i.e., $|N|$).
6. [20 pts.] Below, I give the minimal answer. Note, since this is breadth-first search, the problem said it is okay to stop expanding once goal is generated. However, the official graph-search does the goal test before expanding, so don’t deduct if students generate the additional 8 nodes at level 2, before expanding the goal. Depending on the order of the expansion of children, more states may be generated than shown below. Three points for the initial state, two for each state at level 1, and 1 point for each of the remaining 6 states (a total of 15); 5 points for a correct BFS expansion order.