CS-271, Intro to A.I. — Mid-term Exam — Fall Quarter, 2011

YOUR NAME AND EMAIL ADDRESS: 

YOUR ID: _______  ID TO RIGHT:_______  ROW:____  NO. FROM RIGHT:____

The exam will begin on the next page. Please, do not turn the page until told.

When you are told to begin the exam, please check first to make sure that you have all eight pages, as numbered 1-8 in the bottom-left corner of each page.

The exam is closed-notes, closed-book. No calculators, cell phones, electronics.

Please clear your desk entirely, except for pen, pencil, eraser, an optional blank piece of paper (for optional scratch pad use), and an optional water bottle.

This page summarizes the points available for each question so you can plan your time.

1. (10 pts total, -2 pts for each error, but not negative) MINI-MAX SEARCH IN GAME TREES.

2. (10 pts total, -2 for each error, but not negative) ALPHA-BETA PRUNING.

3. (10 pts total, 2 pts each) CONSTRAINT SATISFACTION PROBLEMS.

4. (10 pts total, 2 pts each) State-Space Search.

5. (15 pts total, 1 pt each) Agent/Search Concepts.

6. (5 pts total, -2 pts each wrong answer, but not negative) Task environment.

7. (20 pts total, 1 pt each) SEARCH PROPERTIES.

8. (10 pts total, -1 each error, but not negative) State Space Search.

9. (10 pts total, -2 each error, but not negative) Optimality of A* Search.
1. (10 pts total, -2 pts for each error, but not negative) **MINI-MAX SEARCH IN GAME TREES.**

1.a. The game tree below illustrates a position reached in the game. It is Max's turn to move. Inside each leaf node is the estimated score of that resulting position returned by the heuristic static evaluator. **FILL IN EACH BLANK SQUARE WITH THE PROPER MINI-MAX SEARCH VALUE.**

![Game Tree Diagram]

1.b. What is the best move for Max? (write A, B, or C) ________

2. (10 pts total, -2 for each error, but not negative) **ALPHA-BETA PRUNING.** The same tree as above. **CROSS OUT EACH LEAF NODE THAT WILL BE PRUNED BY ALPHA-BETA PRUNING.** You do not need to indicate the branch node values again.

![Game Tree Diagram]

***TURN PAGE OVER AND CONTINUE ON THE OTHER SIDE***
3. (10 pts total, 2 pts each) **CONSTRAINT SATISFACTION PROBLEMS.** This problem asks about the Map Coloring Problem. Each region must be colored one of Red (R), Green (G), or Blue (B). Neighboring regions must be a different color. The map (left) and constraint graph (right) are below.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>G</td>
<td>B</td>
<td>R</td>
<td>G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
</tbody>
</table>

**3a. (2 pts total, -1 for each error, but not negative) FORWARD CHECKING.** Consider the partial assignment below. Variable B has been assigned value R as shown. Cross out all values that would be eliminated by Forward Checking (FC) after the assignment to variable B.

```
A B C D E F
R G B R G B R G B R G B
```

**3b. (2 pts total, -1 for each error, but not negative) ARC CONSISTENCY.** Consider the partial assignment below. Variables A and B have been assigned values as shown. Cross out all other values that would be eliminated by Arc Consistency (AC, also called AC-3 in your book).

```
A B C D E F
R G R G B R G B R G B
```

**3c. (2 pts total, -1 for each error, but not negative) MINIMUM-REMAINING-VALUES HEURISTIC.** Consider the partial assignment below. Variable A is already assigned value R, and Arc Consistency is already done. List all unassigned variables that might possibly be selected by the Minimum-Remaining-Values (MRV) Heuristic: ____________________.

```
A B C D E F
R G B R G B G B R G B R G B
```

**3d. (2 pts total, -1 for each error, but not negative) DEGREE HEURISTIC.** Consider the partial assignment below. (It is the same assignment as in problem 3c above.) List all unassigned variables that might possibly be selected by the Degree Heuristic: ____________________.

```
A B C D E F
R G B R G B G B R G B R G B
```

**3e. (2 pts total) MIN-CONFLICTS HEURISTIC.** Consider the complete but inconsistent assignment below. E has just now been selected to be assigned a new value. List all new values that might be chosen below for E by the Min-Conflicts Heuristic? ____________________.

```
A B C D E F
R G B R G B G B R G B R G B
```

```
A B C D E F
R G B R G B G B R G B R G B
```
4. (10 pts total, 2 pts each) **State-Space Search.** Execute Tree Search through this graph (i.e., do not remember visited nodes, so repeated nodes are possible). It is not a tree, but pretend you don’t know that. Step costs are given next to each arc, and heuristic values are given next to each node (as h=x). The successors of each node are indicated by the arrows out of that node. (Note: C is a successor of itself).

For each search strategy below, indicate the order in which nodes are expanded (i.e., to expand a node means that its children are generated), ending with the goal node that is returned as the result from the search. If the search gets stuck in a loop, show the repeating sequence and then “….”

4.a. (2 pts, -1 for each wrong answer, but not negative) **UNIFORM COST SEARCH.**

4.b. (2 pts, -1 for each wrong answer, but not negative) **GREEDY (BEST-FIRST) SEARCH.**

4.c (2 pts, -1 for each wrong answer, but not negative) **ITERATED DEEPENING SEARCH.**

4.d. (2 pts, -1 for each wrong answer, but not negative) **A* SEARCH.**

4.e. (2 pts, -1 for each wrong answer, but not negative) **OPTIMALITY.**

Did Uniform Cost Search find the optimal goal? ______ Why or why not? ____________________________

Did A* Search find the optimal goal? ______ Why or why not? ____________________________

**** TURN PAGE OVER AND CONTINUE ON THE OTHER SIDE ****
5. (15 pts total, 1 pt each) **Agent/Search Concepts.** For each of the following terms on the left, write in the letter corresponding to the best answer or the correct definition on the right. The first one is done for you as an example.

<table>
<thead>
<tr>
<th>A</th>
<th>Agent</th>
<th>A</th>
<th>Perceives environment by sensors, acts by actuators</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Percept</td>
<td>B</td>
<td>All states reachable from the initial state by a sequence of actions</td>
</tr>
<tr>
<td></td>
<td>Performance Measure</td>
<td>C</td>
<td>Guaranteed to find a solution if one is accessible</td>
</tr>
<tr>
<td></td>
<td>Rational Agent</td>
<td>D</td>
<td>Process of removing detail from a representation</td>
</tr>
<tr>
<td></td>
<td>State Space</td>
<td>E</td>
<td>Maximum number of successors of any node</td>
</tr>
<tr>
<td></td>
<td>Search Node</td>
<td>F</td>
<td>Set of all leaf nodes available for expansion at any given time</td>
</tr>
<tr>
<td></td>
<td>Link between nodes</td>
<td>G</td>
<td>Estimates cost of cheapest path from current state to goal state</td>
</tr>
<tr>
<td></td>
<td>Path</td>
<td>H</td>
<td>Guaranteed to find lowest cost among all accessible solutions</td>
</tr>
<tr>
<td></td>
<td>Abstraction</td>
<td>I</td>
<td>Represents a state in the state space</td>
</tr>
<tr>
<td></td>
<td>Optimal Search</td>
<td>J</td>
<td>Sequence of states connected by a sequence of actions</td>
</tr>
<tr>
<td></td>
<td>Complete Search</td>
<td>K</td>
<td>Agent’s perceptual inputs at any given instant</td>
</tr>
<tr>
<td></td>
<td>Expand a state</td>
<td>L</td>
<td>Agent that acts to maximize its expected performance measure</td>
</tr>
<tr>
<td></td>
<td>Frontier</td>
<td>M</td>
<td>Apply each legal action to a state, generating a new set of states</td>
</tr>
<tr>
<td></td>
<td>Search Strategy</td>
<td>N</td>
<td>Represents an action in the state space</td>
</tr>
<tr>
<td></td>
<td>Branching Factor</td>
<td>O</td>
<td>How a search algorithm chooses which node to expand next</td>
</tr>
<tr>
<td></td>
<td>Heuristic Function</td>
<td>P</td>
<td>Evaluates any given sequence of environment states for utility</td>
</tr>
</tbody>
</table>

6. (5 pts total, -2 pts each wrong answer, but not negative) **Task environment.** A task environment is a set of four things, with the acronym PEAS. Fill in the blanks with the PEAS components.

P__________ E__________ A__________ S__________

7. (20 pts total, 1 pt each) **SEARCH PROPERTIES.**

Fill in the values of the four evaluation criteria for each search strategy shown. Assume a tree search where b is the finite branching factor; d is the depth to the shallowest goal node; m is the maximum depth of the search tree; C* is the cost of the optimal solution; step costs are identical and equal to some positive ε; and in Bidirectional search both directions use breadth-first search.

Note that these conditions satisfy all of the footnotes of Fig. 3.21 in your book.

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Complete?</th>
<th>Time complexity</th>
<th>Space complexity</th>
<th>Optimal?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breadth-First</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uniform-Cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Depth-First</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iterative Deepening</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bidirectional (if applicable)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8. (10 pts total, -1 each error, but not negative) **State Space Search.** This is a classic brain-teaser puzzle:

*A man is traveling to market with a fox, a goose, and a bag of oats. He comes to a river. The only way across the river is a boat that can hold the man and exactly one of the fox, goose or bag of oats. The fox will eat the goose if left alone without the man. The goose will eat the oats if left alone without the man. How can the man get all his possessions safely across the river?*

We will call states where something gets eaten “forbidden” and consider them as part of the state space, but from which there is no return; i.e., there is no way to go to any other state (including back to the previous state) from such a forbidden state. All other states are “allowed.” In this state space, an “action” corresponds to the man crossing the river in the boat. The man may cross the river either alone or with at most one of his possessions.

One way to represent this problem is as a state vector with components (M,F,G,O), where M=man, F=fox, G=goose, and O=oats are binary variables that are 0 if the man/fox/goose/oats are on the **near** (close) bank, and 1 if on the **far** (distant) bank. Note that the boat is always on the same side of the river as the man. Thus, the start state is (0,0,0,0), with the man and all his possessions on the **near** bank. The single goal state is (1,1,1,1), with everything on the **far** bank.

8.a. (1 pt) How many total states (both allowed and forbidden) in this state space?

_____________________

8.b. (1 pt) What is the maximum branching factor counting BOTH allowed and forbidden states as children?

_____________________

8.c. (8 pts, -1 for each error, but not negative) Draw the state space considering only ALLOWED states. Represent an allowed state as a circle enclosing its binary vector (M,F,G,O). Connect states that are possible successors of each other. The start and goal states are shown. The first one is done for you as an example.

```
0000    1010    1111
```

**** TURN PAGE OVER AND CONTINUE ON THE OTHER SIDE ****
9. (10 pts total, -2 each error, but not negative) **Optimality of A* Search.** You know that A* tree search with an admissible heuristic is an optimal search strategy. Recall that:

\[
\begin{align*}
g(n) &= \text{true path cost so far} \\
h(n) &= \text{estimated optimal cost from } n \text{ to goal} \leq \text{true optimal cost} \\
f(n) &= g(n) + h(n) = \text{estimated total optimal path cost} \leq \text{true total optimal cost}
\end{align*}
\]

The following is a proof that A* tree search (queue sorted by \(f(n)\)) is optimal if the heuristic is admissible. The lines of the proof are labeled A through G. Unfortunately, they have been mixed up (permuted).

Let \(ng\) be the first goal node popped off the queue. Let \(no\) be any other node still on the queue. We wish to prove that the path through \(no\) can never be extended to a path to any goal node that costs less than the path to \(ng\) that we just found.

A: true total cost of path to \(ng\)
B: \[ \leq f(no) \quad \text{// because queue is sorted by } f \text{ and } ng \text{ came off queue first} \]
C: \[ \leq g(no) + \text{true optimal cost to goal from } no \quad \text{// because } h \text{ is admissible} \]
D: \[ = f(ng) \quad \text{// by definition of } f \text{, with } h(ng) = 0 \text{ because } ng \text{ is a goal node} \]
E: \[ = g(no) + h(no) \quad \text{// by definition of } f \]
F: \[ = g(ng) \quad \text{// because } ng \text{ represents a complete path} \]
G: \[ = \text{true total optimal path cost through } no \]

Fill in the blanks with the letters B, C, D, E, F, and G to prove that the true total cost of the path to \(ng\) is less than or equal to the true total optimal path cost through \(no\). The first and last letters, A and G, have been done for you as an example.

\[ \text{_____ A _____} \text{ _____} \text{ _____} \text{ _____} \text{ _____} \text{ _____} \text{ _____} \text{ _____} \text{ G}. \]