You have 50 minutes to complete this midterm. You are only allowed to use your textbook, your notes, your assignments and solutions to those assignments during this midterm. If you find that you are spending a large amount of time on a difficult question, skip it and return to it when you’ve finished some of the easier questions. Total marks for this midterm is 53.

<table>
<thead>
<tr>
<th>Section</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agents</td>
<td>/ 12</td>
</tr>
<tr>
<td>Logic</td>
<td>/ 19</td>
</tr>
<tr>
<td>Search</td>
<td>/ 10</td>
</tr>
<tr>
<td>Games</td>
<td>/ 12</td>
</tr>
<tr>
<td>Total</td>
<td>/ 53</td>
</tr>
</tbody>
</table>
Section I: Agents (12 points)

1. Minesweeper is a single player game that involves an NxN grid. Certain cells in the grid are mines while other cells are empty. All cells are “covered” initially. A player can uncover a cell by clicking on it. If the player uncovers a mine, the game is over. If the player uncovers an empty square, a number is displayed in the uncovered grid cell showing the number of adjacent cells with mines. The goal is to determine the location of all the mines.

For each part below, circle the choice which best describes the environment for an agent that plays Minesweeper.

a) Fully observable or Partially observable [1 point]  
b) Deterministic or Stochastic [1 point]  
c) Episodic or Sequential [1 point]  
d) Static or Dynamic [1 point]  
e) Discrete or Continuous [1 point]  
f) Single agent or Multi-agent [1 point]  

2. What type of agent was the tic-tac-toe player in Programming Assignment #2? Choose from simple reflex agent, model-based reflex agent, goal-based agent and utility-based agent. Explain your answer. [2 points]

It was either a goal-based agent or a utility-based agent. Technically it was a goal-based agent because you probably didn’t differentiate between the different goal states and you just wanted the agent to win the game.
3. Can you design a simple reflex agent that is rational for the tic-tac-toe domain? Explain your answer. [4 points]

Yes you can do this. You would need a gigantic table that maps every possible tic-tac-toe board to the next move.

II. Propositional Logic [19 points]

1. Suppose you apply resolution to the following two sentences in your KB.
   \[ A \lor \neg B \quad \ldots(1) \]
   \[ \neg B \lor \neg A \quad \ldots(2) \]

   What do you get? Is it the empty clause? Explain your answer. [4 points]
   \[ A \lor \neg A = \text{true} \]

   This is not the same as the empty clause (which is the false value). Since we obtain true, it means that the KB is a tautology, meaning the KB is useless.

2. The following questions deal with the statements below:
   Either Bob is insane or (he is a genius and an AI Researcher)
   If Bob is a genius, then he is not evil.
   If Bob is insane, then he is evil
   Bob is not a genius or he is not an AI Researcher

   a) Convert the English sentences above into propositional logic. [5 points]
   \[ \text{Insane} \lor (\text{Genius} \land \text{AIResearcher}) \]
   \[ \text{Genius} \implies \neg \text{Evil} \]
   \[ \text{Insane} \implies \text{Evil} \]
   \[ \neg \text{AIResearcher} \lor \neg \text{Genius} \]

   b) Convert the propositional logic sentences from part (a) into CNF. [5 points]
   \[ \text{Insane} \lor (\text{Genius} \land \text{AIResearcher}) \]
   \[ = (\text{Insane} \lor \text{Genius}) \land (\text{Insane} \lor \neg \text{AIResearcher}) \]
Genius \Rightarrow \neg Evil
= \neg Genius \lor \neg Evil

Insane \Rightarrow Evil
= \neg Insane \lor Evil

\neg AIResearcher \lor \neg Genius

c) Does KB \models (Bob is evil)? Show the resolution steps if it does. Otherwise, explain why it does not resolve. [5 points]

\begin{align*}
\text{Insane} \lor \text{Genius} & \quad \text{...(1)} \\
\text{Insane} \lor \text{AIResearcher} & \quad \text{...(2)} \\
\neg \text{Genius} \lor \neg \text{Evil} & \quad \text{...(3)} \\
\neg \text{Insane} \lor \text{Evil} & \quad \text{...(4)} \\
\neg \text{AIResearcher} \lor \neg \text{Genius} & \quad \text{...(5)} \\
\neg \text{Evil} & \quad \text{...(6)}
\end{align*}

Resolve (6) and (4)
\neg \text{Insane} \quad \text{...(7)}

Resolve (7) and (2)
\text{AIResearcher} \quad \text{...(8)}

Resolve (5) and (8)
\neg \text{Genius} \quad \text{...(9)}

Resolve(9) and (1)
\text{Insane} \quad \text{...(10)}

Resolve (10) and (7)
{} \quad \text{...(11)}

Therefore, KB \models (Bob is evil)
III. Search [10 points]
1. The resolution refutation algorithm (this was the algorithm we covered in class that uses resolution to complete a proof-by-contradiction) can determine if $ KB \vdash \alpha,$ provided the KB and the query $\alpha$ are in propositional logic. The resolution refutation algorithm can be modeled as a search problem.

a) If we formulate the resolution refutation algorithm as a search problem, what does a state in this search space look like? [2 points]

A state is the current list of sentences in the KB, including any sentences that were deduced.

b) What is the initial state for the resolution refutation search problem? [2 points]
The initial state is the KB with $\neg \alpha$

c) What is the goal state for the resolution refutation search problem? [2 points]
The empty state $\{}$ i.e. the false statement

d) What is the successor function for the resolution refutation search problem? [4 points]
The successor is an application of resolution to two sentences in the KB that produces a new sentence which is then added to the KB.
IV. Games [12 points]

1. Evaluation functions try to approximate the expected utility of the game at a non-terminal node. One problem with them is that they may not be a good approximation. Suppose I create a perfectly accurate evaluation function that generates the remainder of the search tree from the non-terminal node and computes its exact minimax value. Is this a good idea? Why or why not? [2 points]

This is a terrible idea because it defeats the purpose of having an evaluation function. The goal of an evaluation function is to provide a quick, approximate estimate of the utility of a non-terminal node without having to store the entire game tree in memory.

2. Compute the expectiminimax value of the root node below. Show your intermediate work for partial credit. [10 points]

```
Max

<table>
<thead>
<tr>
<th>Chance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chance</td>
</tr>
<tr>
<td>P=0.1</td>
</tr>
<tr>
<td>P=0.5</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>-80</td>
</tr>
</tbody>
</table>

| P=0.9 |
| 2     |

<table>
<thead>
<tr>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
</tr>
</tbody>
</table>

| 4    |
| 6    |
| -9   |
| -5   |
```

2

\[(0.1)(10) + (0.9)(2) = 0.1 + 1.8 = 1.9\]

\[(0.5)(100) + (0.5)(-80) = 50 - 40 = 10\]

\[(0.1)(10) + (0.9)(2) = 0.1 + 1.8 = 1.9\]

\[(0.5)(100) + (0.5)(-80) = 50 - 40 = 10\]