Write your answers on these pages and show your work. If you feel that a question is not fully specified, state any assumptions you need to make in order to solve the problem. You may use the backs of these sheets for scratch work.

Write your name on this and all other pages of this exam. Make sure your exam contains six problems on six pages.

Name

________________________________________________

Student ID

________________________________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>_____</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>_____</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>_____</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>_____</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>_____</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>_____</td>
<td>12</td>
</tr>
<tr>
<td>Total</td>
<td>_____</td>
<td>100</td>
</tr>
</tbody>
</table>

(over)
PROBLEM 1 - Search (30 points)
Consider the search graph drawn below; the start and goal states are labeled. Note that arcs are directed. For each of the search strategies listed below, indicate which goal state is reached (if any) and list, in order, the states expanded. (A state is expanded when it is removed from the OPEN list.) When all else is equal, nodes should be expanded in alphabetical order.

**Depth-First Search**
Goal state reached: _____ States expanded: ____________________________________

**Iterative Deepening**
Goal state reached: _____ States expanded: ____________________________________

**Best-First Search (using \( f = g + h \))**
Goal state reached: _____ States expanded: ____________________________________

**Hill Climbing (using the \( h \) function only)**
Goal state reached: _____ States expanded: ____________________________________

**Beam Search (with a beam width of 2)**
Goal state reached: _____ States expanded: ____________________________________

Would the \( h \) function in this graph lead to an admissible search? _____
Explain your answer:

(over)
PROBLEM 2 - Forward-Chaining: Production Systems (22 points)

(a) Consider the following rule base for a production system:

(1) \( p(X) \land q(Y) \land r(X,Y) \rightarrow assert \ s(X) \)
(2) \( p(X) \land q(X) \land s(X) \rightarrow retract \ s(X) \)
(3) \( r(X,Y) \rightarrow assert \ s(X) \land retract \ p(X) \land retract \ q(Y) \)

Assume working memory (WM) is currently of the form:

\( p(1) \land p(2) \land q(1) \land q(2) \land r(1,2) \land r(2,1) \land r(2,2) \)

Finally, assume that the conflict-resolution scheme used is the same one used in HW5 (i.e., pick the top-most rule whose preconditions are satisfied; however, a given rule cannot be used twice with the same bindings).

Show below the first five steps this production system would perform (even though you only need to find the top-most acceptable rule, you must still show the full conflict set):

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Conflict Set</th>
<th>Rule Fired</th>
<th>Changes to WM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Imagine you wish to create an “email agent” by writing a production system that preprocesses your incoming email. For simplicity, assume someone gives you a program that parses your message into an FOPC representation along the lines of:

\( from(smith) \land to(jones) \land to(lee) \land cc(starr) \land subject("today’s meeting") \land ... \)

Write two useful production rules for this task. Explain what these rules do. (The two rules should have reasonably different preconditions and effects.)

(1)

(2)
PROBLEM 3 - Backward-Chaining: Prolog (18 points)

a) Consider the following Prolog rulebase:

```
p(1,2).
p(3,3).
p(2,1).
q(1,2).
q(2,1).
q(2,2).
q(3,3).
r(2,1).
r(2,2).
s(X1,Y1) :- p(X1,Y1), q(Y1,X1), r(X1,X1).
s(X2,Y2).
r(X3,Y3) :- q(X3,Y3).
w(X4,Y4) :- p(X4,Y4), q(2,Z4), r(X4,Z4).
```

For the following queries, report the answers that Prolog a requested answer. Report fail if Prolog would not be able to find a requested answer.

i) \(?- s(2,1).\)

First Binding List Found: _____________________________
Second Binding List (if one): _____________________________
Third Binding List (if one): _____________________________

ii) \(?- w(U,V).\)

First Binding List Found: _____________________________
Second Binding List (if one): _____________________________
Third Binding List (if one): _____________________________

b) Write a Prolog program for $sisters(X,Y)$, which determines if $X$ and $Y$ are sisters. You may only assume that the predicates $grand_parent(X,Y)$, $parent(X,Y)$, $ancestor(X,Y)$, $male(X)$, $female(X)$, $tall(X)$, $young(X)$, $older_than(X,Y)$, and $different(X,Y)$ are defined. $Different(X,Y)$ is true when $X$ and $Y$ have been bound to different constants; the semantics of the other predicates should be obvious (the binary ones, $predicate(X,Y)$, can be read “$X$ is a predicate of $Y$”).

(over)
PROBLEM 4 - Important AI Concepts (12 points)
In the space below, provide brief and succinct summaries of the importance of the following AI concepts.

________________________

horizon effect

________________________

interacting subgoals

________________________

negation-by-failure

PROBLEM 5 - Game Playing (6 points)
Assume that a game has a static-board evaluator (SBE) that only returns one of three values: -1 (lose), 0 (draw), and 1 (win). Consider the game tree drawn below. What is the fewest possible number of calls to the SBE? To illustrate your answer, label those leaf nodes that would have to be called with the score that the SBE would have to return. (If you feel there are several way that the fewest calls could occur, you need only show one of them.)

(over)
PROBLEM 6 - Lisp (12 points)
The function defined below, \( \text{mark-doubles } X \ Y \), is supposed to replace, by the atom \text{twice}, all those sexpr\’s in list \( X \) that are also in the \textit{top-level} of list \( Y \). That is, it should operate as follows:

\[
\begin{align*}
> & (\text{mark-doubles '}(1 2 3) ' (a b c)) \\
& (1 2 3)
\\
> & (\text{mark-doubles '}(1 (2) 3) ' (3 this list sure is long (1) 2)) \\
& (1 (\text{twice} twice)
\\
> & (\text{mark-doubles '}(1 (hi there) 3) ' (hi there)) \\
& (1 (\text{twice twice} 3)
\\
> & (\text{mark-doubles '}(1 (hi there) 3) ' (an aloha (hi there) to you)) \\
& (1 twice 3)
\end{align*}
\]

---

(defun mark-doubles (X Y)
 "This code is buggy."
 (cond ((member X Y)

   twice)

   ((atom X)

   nil)

   (t (dolist (item X)

       (mark-doubles item Y)))

   )
)

Your task is to debug this function. To the right of the code, describe the errors in it and show the necessary corrections.