NAME:_____ **SID#:**_____ **Login:**_____ **Sec:**_____

CS 188 Fall 2006 Introduction to Artificial Intelligence Midterm

You have 80 minutes. The exam is closed book, one page cheat sheet allowed, non-programmable basic calculators allowed. 80 points total. Pace yourself, and don't panic!

Mark your answers ON THE EXAM ITSELF. Write your name, SID, login, and section number at the top of each page.

If you are not sure of your answer you may wish to provide a *brief* explanation. All short answers can be successfully answered in a few sentences *at most*.

For staff use only

Q. 1	Q. 2	Q. 3	Q. 4	Q. 5	Q. 6	Total
/17	/10	/15	/12	/16	/10	/80

1. (17 points.) Short Answer

Answers should fit in a single sentence.

(a) **(3 pts):** Give a set of broad conditions under which A* search reduces to BFS.

(b) **(3 pts):** When would DFS be a better choice than A* search? Describe a broad qualitative condition rather than a specific example.

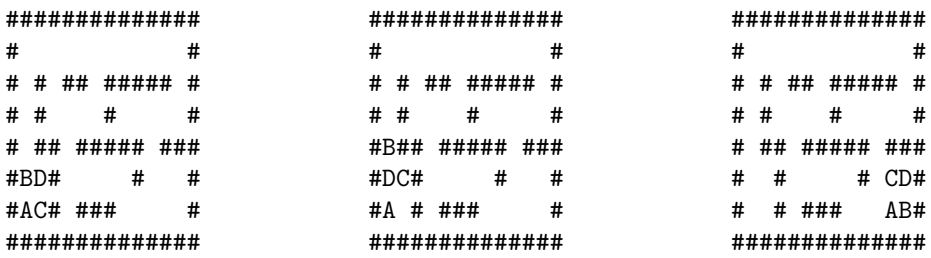
(c) **(3 pts):** Why is temporal difference learning of q-values (Q-learning) superior to temporal difference learning of values?

(d) **(3 pts):** Write a Bellman update for *q-value iteration*, which is like value iteration except q-values rather than values are learned from previous q-value estimates, using a one-step lookahead (i.e. you should express Q_{i+1} estimates in terms of Q_i estimates).

(e) **(5 pts):** In a *Markov game*, or adversarial MDP, two players, MAX and MIN alternate actions in an MDP. Assume the game is zero-sum, so that when a transition (s, a, s') occurs, MAX receives $R(s, a, s')$, while MIN receives $-R(s, a, s')$, regardless of who initiated the transition. Assume that both players have the same set of actions available in any state and that both players use the same discount per time step. Let $V_{MAX}(s)$ and $V_{MIN}(s)$ be the expected future discounted rewards for each player. Write *two* Bellman equations, expressing each of V_{MAX} and V_{MIN} in terms of adjacent lookahead values of V_{MAX} and / or V_{MIN}.

2. (10 points.) Search and Heuristics

Consider the following variant of Mazeworld, shown below, where a *single planning agent* must move four robots from a configuration on one side of the grid to a goal configuration on the other side, shown below. In any given time step, each robot can move one square North, South, East, West, or stay where it is. After the moves occur, no two robots can occupy the same result square. However, two adjacent robots *can* swap positions in a single time step. A possible successor of the start state is shown below in which B moved North, D moved West, C moved North, and A stood still.



(1) (2 pts): What is the *branching factor* for this search problem. Give the tightest (smallest) upper bound you can.

(2) (2 pts): If the grid is n by n, what is the size of the *state space*? Give the tightest upper bound you can.

(3) (4 pts): Propose an efficient, simple admissible heuristic for use in A* search. The number of computations made by your heuristic should not depend on the size of the grid. Trivial and nearly trivial answers (e.g. "1" unless at a goal state) will not receive credit.

(4) (2 pts): Propose an admissible heuristic that takes advantage of the functionality of your Mazeworld code. Your heuristic should return the correct forward cost (i.e. be the perfect heuristic) in the case of a single robot, but need not be constant time.

3. (15 points.) CSPs

You are in charge of scheduling for computer science classes that meet Mondays, Wednesdays and Fridays. There are 6 classes that meet on these days and 3 professors who will be teaching these classes. You are constrained by the fact that each professor can only teach one class at a time.

The classes are:

- Class 1 - Intro to Programming: meets from 8:00-9:00am
- Class 2 - Intro to Artificial Intelligence: meets from 8:30-9:30am
- Class 3 - Natural Language Processing: meets from 9:00-10:00am
- Class 4 - Computer Vision: meets from 9:00-10:00am
- Class 5 - Machine Learning: meets from 10:30-11:30am

The professors are:

- Professor A, who is available to teach Classes 1, 2, and 5.
- Professor B, who is available to teach Classes 3, 4, and 5.
- Professor C, who is available to teach Classes 1, 3, and 4.

(1) (4 pts): Formulate this problem as a CSP problem in which there is one variable per class, stating the domains, and constraints. Constraints should be specified formally and precisely, but may be implicit rather than explicit.

(2) (2 pts): Draw the constraint graph associated with your CSP.

(4) (4 pts): Show the domains of the variables after running arc-consistency on this initial graph (after having already enforced any unary constraints).

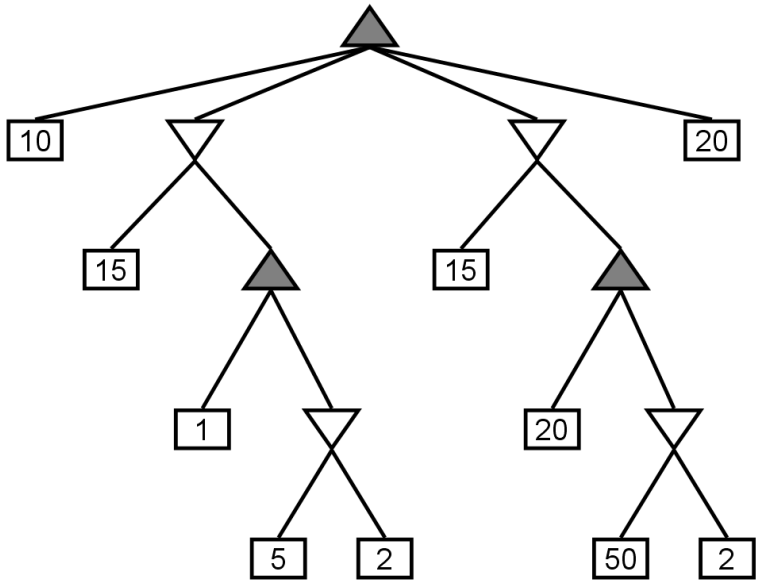
(5) (1 pt): Give one solution to this CSP.

(6) (2 pts): Your CSP should look nearly tree-structured. Briefly explain (one sentence or less) why we might prefer to solve tree-structures CSPs.

(7) (2 pts): Name (or briefly describe) a standard technique for turning these kinds of nearly tree-structured problems into tree-structured ones.

4. (12 points.) Minimax Search

Consider the following minimax tree.



(1) (2 pts): What is the minimax value for the root?

(2) (5 pts): Draw an X through any nodes which will not be visited by alpha-beta pruning, assuming children are visited in left-to-right order.

(3) (2 pts): Is there another ordering for the *children of the root* for which more pruning would result? If so, state the order.

(4) (3 pts): Propose a general, practical method for ordering children of nodes which will tend to increase the opportunities for pruning. You should be concise, but clearly state both what to do about min nodes and max nodes.

5. (16 points.) Multi-Agent Search

(1) (4 pts): For the case where ghosts move randomly, state Pacman (with Project 1 rules) as an MDP. Each component should be answered in *at most* one sentence:

States: Configurations of the board, as given in a GameState object, including food locations, agent positions, etc.

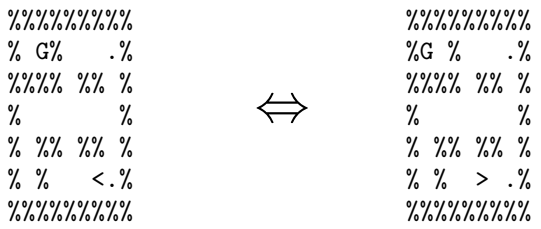
Actions: A subset of {*north, south, east, west, stop*}, varying by state.

Transitions:

Rewards:

Discount:

Answer the following questions precisely, but in around one sentence.



(2) (3 pts): Pacman thrashes, alternating between the two configurations shown (with some irrelevant motion from the ghost), making no progress. If Pacman is using minimax search, depth 3, with the state score as the evaluation function, why would he exhibit this behavior?

(3) (2 pts): Would an increase to the depth fix this? If so, what depth would be needed for Pacman to eat the dot and move on? If not, why not?

(4) (2 pts): Keeping the depth at 3, how could we fix this thrashing with a basic idea from MDPs?

(5) (5 pts): If Pacman knows that the ghosts are using 2-ply minimax, with a known evaluation function, what is his optimal search process? You may assume that there is only one ghost. Clearly specify an algorithm which describes the computation you recommend.

6. (10 points.) MDPs and Reinforcement Learning

Consider an autonomous robot which can either move FAST or SLOW in any time step. Moving FAST generally gives a reward of +2, while moving SLOW gives a reward of only +1. However, the robot must also take into account its internal temperature, which can be either HOT or OK. Driving SLOW tends to lower the temperature, while driving FAST tends to raise it. If the robot is HOT, there is a danger if it overheating, at which point it must stop, cool down, and make repairs. The MDP transitions and rewards are specified as follows:

s	a	s'	$T(s,a,s')$	$R(s,a,s')$
OK	SLOW	OK	1.0	+1
OK	FAST	OK	0.5	+2
OK	FAST	HOT	0.5	+2
HOT	SLOW	OK	1.0	+1
HOT	FAST	HOT	0.5	+2
HOT	FAST	OK	0.5	-10

Note that while repairs are costly, the robot is OK afterwards (the last row in the table).

(1) (5 pts): Run two rounds of value iteration in the table below, using a discount of 0.8. You may skip the greyed-out square.

s	V_0	V_1	V_2
OK	0		
HOT	0		

(1) (5 pts): Run Q-learning with a discount of 0.8 and a learning rate of 0.5, using the transition samples below. Do not copy over q-values which have not changed in a given step.

Assume the agent experiences the samples:

OK, FAST, HOT, reward +2, calculate Q_1
HOT, FAST OK, reward -10, calculate Q_2
OK, SLOW, OK, reward +1, calculate Q_3

s	a	Q_0	Q_1	Q_2	Q_3
OK	SLOW	0			
OK	FAST	0			
HOT	SLOW	0			
HOT	FAST	0			

End of Exam