CS 188
Fall 2022

Introduction to Artificial Intelligence

Midterm

- You have 110 minutes.
- The exam is closed book, no calculator, and closed notes, other than one double-sided cheat sheet that you may reference.
- For multiple choice questions,

  □ means mark **all options** that apply

  ○ means mark a **single choice**

First name	
Last name	
SID	
Name and SID of person to the right	
Name and SID of person to the left	
Discussion TAs (or None)	

Honor code: "As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others."

By signing below, I affirm that all work on this exam is my own work, and honestly reflects my own understanding of the course material. I have not referenced any outside materials (other than one double-sided cheat sheet), nor collaborated with any other human being on this exam. I understand that if the exam proctor catches me cheating on the exam, that I may face the penalty of an automatic "F" grade in this class and a referral to the Center for Student Conduct.

Signature: ______________________

Point Distribution

Q1.	Potpourri	22
Q2.	Haunted House	17
Q3.	Rocket Science	16
Q4.	Golden Bear Years	13
Q5.	Games	8
Q6.	Indiana Jones & the Kingdom of the Crystal Skull	12
Q7.	Reinforcement Learning	12
	Total	100

Q1. [22 pts] Potpourri

(a) Evgeny has an arbitrary search problem. Assume all costs are positive.

(i) [1 pt] If he runs uniform cost graph search on the problem, he is guaranteed to get the optimal solution.

○ True ○ False

(ii) [1 pt] Jason takes the problem and designs a heuristic function h such that for all states s, $h(s) > 0$. If he runs greedy graph search on the problem using the heuristic, he is guaranteed to get the optimal solution.

○ True ○ False

(iii) [1 pt] Regardless of your answer for the previous parts, suppose both uniform cost graph search and greedy graph search return the same optimal path. Jason claims that using his heuristic, A* graph search is guaranteed to return the optimal path for Evgeny's search problem. Is he correct?

○ Yes, because the priority value for A* search on a given state is the sum of priority values for UCS and greedy search.

○ Yes, but not for the reason above.

○ No, because if the heuristic is not consistent, then A* graph search is not guaranteed to be optimal.

○ No, but not for the reason above.

(b) [2 pts] In a CSP, what could happen to the domain of an arbitrary variable after enforcing arc consistency? Select all that apply.

□ The domain could remain unchanged.

□ The domain size could be smaller than before (but not empty).

□ The domain could be empty.

○ None of the above

(c) Consider a general CSP with n variables of domain size d. We would like to use a search tree to solve the CSP, where all the complete assignments (and thus all the solutions) are leaf nodes at depth n.

(i) [2 pts] How many possible complete assignments are there for this CSP?

○ $O(n^2d^3)$

○ $O(n \cdot d)$

○ $O(n^2)$

○ $O(n^2d)$

○ $O(n^d)$

○ $O(d^n)$

○ None of the above

(ii) [2 pts] How many leaves does the search tree have?

○ $O(n^2 \cdot d^n)$

○ $O(d! \cdot n^d)$

○ $O(d^2 \cdot n^d)$

○ $O(n! \cdot d^n)$

○ Same as the answer to the previous subpart

○ None of the above

(d) Suppose you are playing a zero-sum game in which the opponent plays optimally. There are no chance nodes in the game tree.

(i) [1 pt] If you want to maximize your utility when playing against this opponent, which algorithm would be the best fit? Assume all the algorithms don't have a limit on search depth.

○ Minimax

○ Expectimax

○ Monte Carlo Tree Search

○ Not enough information

(ii) [2 pts] Select all true statements about pruning game trees.

□ Using alpha-beta pruning allows you to choose an action with a higher value compared to not using alpha-beta pruning.

□ Alpha-beta pruning ensures that all the nodes in the game tree have their correct values.

□ Alpha-beta pruning will always prune at least one node/leaf in a minimax game tree.

□ Alpha-beta pruning does not work on expectimax game trees with unbounded values.

○ None of the above

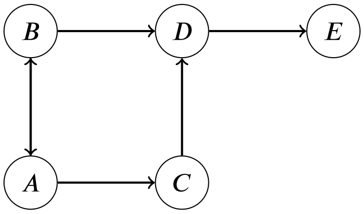
(e) [2 pts] Select all true statements about alpha-beta pruning.

□ Alpha-beta pruning affects the action selected at the root node.
□ The order in which nodes are pruned affects the number of nodes explored.
□ The order in which nodes are pruned affects the action selected at the root node.
□ In most cases, pruning reduces the number of nodes explored.
○ None of the above

(f) [2 pts] Select all true statements about minimax, minimax with alpha-beta pruning, and Monte Carlo tree search (MCTS).

□ Among the three search algorithms, only MCTS is a sample-based search algorithm.
□ Neither minimax with alpha-beta pruning nor MCTS will ever search the tree exhaustively (i.e. visit every node in the search tree).
□ Minimax is an exhaustive search algorithm (i.e. it visits every node in the search tree).
□ When the game tree is small, minimax can explore fewer nodes than minimax with alpha-beta pruning.
○ None of the above

(g) The following graph defines an MDP with deterministic transitions. Each transition produces a constant reward $r = 1$. The discount factor is $\gamma = 0.5$. The game ends once the state E is reached (there are no actions available from E).



Note: The formula for the sum of an infinite geometric series is $\sum_{i=0}^{\infty} x^i = 1 + x + x^2 + x^3 + ... = \frac{1}{1-x}$

(i) [1 pt] What is $V^*(D)$, the optimal value at state D?

□

(ii) [1 pt] What is $V^*(A)$, the optimal value at state A?

□

(iii) [1 pt] How many iteration(s) are needed for policy iteration to converge in the **worst case** (for any initial policy π_0)? In other words, select the minimum k such that $\pi_k = \pi^*$ for the worst π_0.

○ 0
○ 1
○ 2
○ 3
○ ∞ (never converges in the worst case)
○ None of the above

(iv) [1 pt] How many iteration(s) are needed for value iteration to converge if $V_0(s) = 0$ for all states s? In other words, select the minimum k such that $V_k(s) = V^*(s)$.

○ 0
○ 1
○ 2
○ 3
○ ∞ (never converges)
○ None of the above

(h) [2 pts] Select all true statements about reinforcement learning.

□ Direct Evaluation requires knowing the transition function of the underlying MDP.
□ TD Learning itself does not learn the optimal value/policy.
□ The optimal Q-value $Q^*(s, a)$ is the expected discounted reward for following the optimal policy starting at state s.
□ Q-learning can learn the optimal policy using only transition data from random policies.
○ None of the above

Q2. [17 pts] Haunted House

Mr. and Mrs. Pacman have found themselves inside of a haunted house with H floors. Each floor is a rectangular grid of dimension $M \times N$. Mr. Pacman and Mrs. Pacman have been separated, and must find each other. There is one staircase per floor, all located on the same square in the $M \times N$ grid. At each timestep, Mr. Pacman or Mrs. Pacman can move *North*, *East*, *South*, *West*, *Up*, or *Down* (they can take *Up* or *Down* only if they are at a staircase). However, Mr. Pacman can only travel L levels of stairs before he has to rest for W turns. Mr. and Mrs. Pacman alternate turns.

(a) **(i)** [1 pt] What is the maximum branching factor?

□

(ii) [2 pts] Assume that all actions have a cost of 1, except for *Up* and *Down*, which have a cost of 2. Which of the following search algorithms will be guaranteed to return the solution with the fewest number of actions? Select all that apply.

□ Depth-First Search　　□ Uniform-Cost Search

□ Breadth-First Search　　○ None of the above

(iii) [6 pts] For **a(iii) and a(iv)** only, assume there are K knights that Mr. and Mrs. Pacman have to avoid, located throughout the house. Since the knights are wearing heavy armor, they cannot move. Fill in the blanks such that S evaluates to the minimal state space size. Each blank can include numbers (including 0) or variables in the problem statement.

$$S = A \cdot 6^{B} \cdot M^{C} \cdot N^{D} \cdot H^{E} \cdot K^{F}$$

A = □ B = □ C = □ D = □ E = □ F = □

(iv) [2 pts] Assume that all actions have a cost of 1, except for *Up* and *Down*, which have a cost of 2. Which of the following search algorithms will be guaranteed to return the solution with the fewest number of actions? Select all that apply.

□ Depth-First Search　　□ Uniform-Cost Search

□ Breadth-First Search　　○ None of the above

(b) For each modification to the original problem, **write the term X such that the new minimal state space size S' is $X \cdot S$.**

Each subpart below is **independent** (the modifications are not combined).

(i) [2 pts] Mr. and Mrs. Pacman discover two elevators that start on floor 1 and floor H respectively. On each turn, each elevator moves one level up and down, respectively. When an elevator reaches the top or bottom floor, it reverses direction. This motion is repeated for all time steps.

□

(ii) [2 pts] Mr. and Mrs. Pacman find out that there are indistinguishable ghosts inside the house. There are G ghosts, where **G is much smaller than $M \cdot N \cdot H$**. The ghosts move all at once randomly after each turn, and there can only be one ghost in a single square.

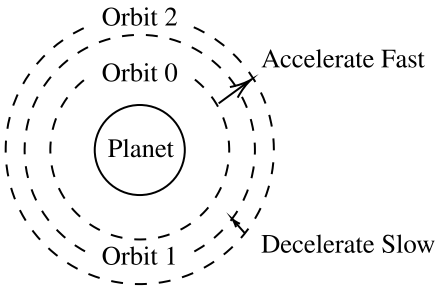
□

(iii) [2 pts] Same as the previous subpart, but now there are more ghosts.

Specifically, $G > M \cdot N \cdot H \cdot \frac{\log(2)}{\log(M \cdot N \cdot H)}$. (You don't need this expression to solve this problem.)

Q3. [16 pts] Rocket Science

We are in a spaceship, orbiting around a planet. The actions available in this search problem are listed below:



Action	Result	Cost
Accelerate Fast	Move up 2 orbits	$5 + 3k$
Accelerate Slow	Move up 1 orbit	$5 + k$
Decelerate Fast	Move down 2 orbits	$5 + 3k$
Decelerate Slow	Move down 1 orbit	$5 + k$

For all subparts, assume that we are performing A* search, and **a solution exists from any state**. Select whether the provided heuristic is admissible and whether the provided heuristic is consistent, **for any choice of** $k > 0$.

(a) Suppose our goal is a specific orbit.

(i) [2 pts] h_1 = min # of actions needed to go to the goal orbit if **only Slow actions** are allowed

□ Admissible　　□ Consistent　　○ Neither

(ii) [2 pts] $h_2 = h_1 \cdot (5 + k)$

□ Admissible　　□ Consistent　　○ Neither

(iii) [2 pts] $h_3 = \min\,[h_1 \cdot (5 + k),\ m]$, $\quad m = \begin{cases} 0.5 \cdot h_1 \cdot (5 + 3k) + 0.5 \cdot (5 - k) & \text{if } h_1 \text{ is odd} \\ 0.5 \cdot h_1 \cdot (5 + 3k) & \text{otherwise} \end{cases}$

□ Admissible　　□ Consistent　　○ Neither

(iv) [2 pts] Which of the following heuristics will find the optimal solution the fastest?

○ h_1　　○ h_2　　○ h_3　　○ None of the above

(b) Now, suppose there are many planets in space. Each planet can have a different number of orbits. Our goal is a specific orbit of a specific planet.

Now, the only actions available are {*Accelerate Slow*, *Decelerate Slow*, *Transition*}. The spaceship can only transition to a different planet from the current planet's outermost orbit, and it will land in the outermost orbit of the destination planet. The *Transition* action has a cost of $5 + 2k$.

(i) [2 pts] $h_4 = \begin{cases} 0 & \text{if orbiting the correct planet} \\ 1 & \text{otherwise} \end{cases}$

□ Admissible　　□ Consistent　　○ Neither

(ii) [2 pts] $h_5 = h_4 \cdot$ (min # of actions needed to go to the outermost orbit of the current planet)

□ Admissible　　□ Consistent　　○ Neither

(iii) [2 pts] $h_6 = (1 - h_4) \cdot$ (min # of actions needed to go to the goal orbit)

□ Admissible　　□ Consistent　　○ Neither

(iv) [2 pts] $h_7 = \min(h_5, h_6)$

□ Admissible □ Consistent ○ Neither

Q4. [13 pts] Golden Bear Years

Pacman will be a freshman at University of Perkeley, so he is planning his 4 years in college. He loves playing video games, so he will only choose courses from the computer science department: $CS_1, CS_2, \dots, CS_8$.

In this problem, the variables are the 8 courses, and their domains are the 4 years to take them. The constraints are listed below:

- CS_1 and CS_2 should be taken in Year 1.
- CS_3 should be taken in Year 3.
- CS_5 should be taken in Year 4.
- CS_5 and CS_6 should be taken in the same year.
- CS_5 and CS_7 should be taken in adjacent years, but it doesn't matter which course is taken first.
- CS_7 and CS_8 shouldn't be taken in the same year.

(a) [1 pt] How many binary constraint(s) are there in the above problem?

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

(b) [8 pts] Pacman has to take all 8 courses in 4 years, so he decides to run backtracking search with arc consistency. Select the values in the domains that will be **removed** after enforcing **unary constraints** and **arc consistency**. If no values are removed from the domain, select "No values removed" (write N).

(1) CS_1: □ 1 □ 2 □ 3 □ 4 ○ N No values removed

(2) CS_2: □ 1 □ 2 □ 3 □ 4 ○ N No values removed

(3) CS_3: □ 1 □ 2 □ 3 □ 4 ○ N No values removed

(4) CS_4: □ 1 □ 2 □ 3 □ 4 ○ N No values removed

(5) CS_5: □ 1 □ 2 □ 3 □ 4 ○ N No values removed

(6) CS_6: □ 1 □ 2 □ 3 □ 4 ○ N No values removed

(7) CS_7: □ 1 □ 2 □ 3 □ 4 ○ N No values removed

(8) CS_8: □ 1 □ 2 □ 3 □ 4 ○ N No values removed

(c) [2 pts] Now, suppose University of Perkeley has provided n total CS courses (n is much larger than 8). Pacman doesn't worry about graduation, so he can spend d years at the school (d is much larger than 4). After taking the prerequisite courses (CS_1 to CS_8), the school has no other constraints on the courses Pacman can take. Now, what will be the time complexity of the AC-3 arc consistency algorithm?

○ The time complexity is $O(n^2d^3)$ and cannot be further reduced.
○ The time complexity is generally $O(n^2d^3)$, and can be reduced to $O(n^2d^2)$.
○ The time complexity can be less than $O(n^2d^2)$.

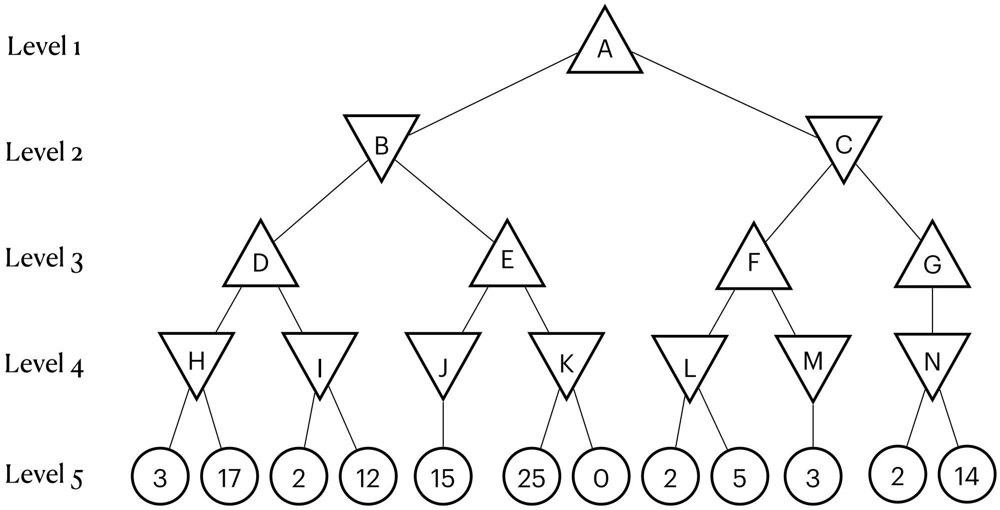
(d) [2 pts] Suppose that Pacman is running local search to find a solution to this CSP, and currently has the following variable assignment. For the next iteration, he wants to change the assignment of one variable, which will lead to the fewest number of unsatisfied constraints remaining. Fill in the blank for the variable and the value it should change to.

$CS_1 : 3 \quad CS_2 : 2 \quad CS_3 : 4 \quad CS_4 : 4$
$CS_5 : 4 \quad CS_6 : 1 \quad CS_7 : 2 \quad CS_8 : 4$

The variable: □ will be assigned to the value: □

Q5. [8 pts] Games

(a) Consider the following game tree.



(i) [1 pt] What is the minimax value at node A?

(ii) [3 pts] Which branches will be pruned after running minimax search with alpha-beta pruning? For instance, if the edge between node A and node B is pruned, write $A - B$. If the edge between H and 3 is pruned, write $H - 3$. List the pruned branches from left to right. If a branch from an upper level is pruned, you don't have to list the branches below that.

(iii) [2 pts] Suppose all the min nodes in layer 4 are changed to max nodes and all the max nodes in layer 3 are changed to min nodes. In other words, we have max, min, min, max as levels 1, 2, 3, 4 in the game tree. We then run minimax search with alpha-beta pruning.

Which of the following statements is true?

○ The same set of leaf nodes will be pruned, because this is still a minimax problem and running alpha-beta pruning will result in the same set of leaf nodes to be pruned.

○ A different set of leaf nodes will be pruned.

○ No leaf nodes can be pruned, because pruning is only possible when the minimizer and maximizer alternate at each level.

○ None of the above

(iv) [2 pts] Now, consider another game tree which has the same structure as the original game tree shown above. This modified game tree can take on any values at the leaf nodes. What is the minimum and the maximum number of leaf nodes that can be pruned after running alpha-beta pruning?

Minimum leaf nodes pruned: □ Maximum leaf nodes pruned: □

Q6. [12 pts] Indiana Jones & the Kingdom of the Crystal Skull

Help Indiana Jones find the crystal skull that was hidden somewhere at the old frozen lake!

For all parts of the problem, assume that value iteration begins with all states initialized to zero, and $\gamma = 1$.

(a) Suppose that we are performing value iteration on the grid world MDP below. Indiana starts at the bottom-left part of the lake labeled 1.

Indiana found an ancient manuscript detailing the rules of this MDP. **(All of these rules are also described in the transition table below.)**

- Indiana can only go up or right, unless the action would cause Indiana to move off the board or into the black square.
- Because the lake is frozen, if Indiana moves up from Square 3, it's possible (with probability x) that Indiana slips and actually moves two squares up.
- From Squares 1, 2, and 4, Indiana deterministically moves one square in the chosen direction.
- Once Indiana reaches Square 5, the only action available is Exit, which gives reward 100. (There are no actions available after exiting.)
- Square 2 contains ancient money, so any action that moves Indiana into Square 2 gives reward +5.
- All other transitions give reward -4 if Indiana moves two squares, and -10 if Indiana moves one square.

■	5 Crystal Skull
2 Ancient Money + 5 reward	4
1 Start	3

s	a	s'	$T(s, a, s')$	$R(s, a, s')$
1	Up	2	1.0	+5
1	Right	3	1.0	-10
2	Right	4	1.0	-10
3	Up	4	$1 - x$	-10
3	Up	5	x	-4
4	Up	5	1.0	-10
5	Exit	End	1.0	100

(i) [2 pts] Find the optimal state values for Square 2 and Square 3. Your answer can be in terms of x.

$V^*(2)$: □ $V^*(3)$: □

(ii) [2 pts] For which values of x would Indiana prefer to go right on Square 1?

□ $< x$

(b) [6 pts] The rest of this question is independent of the previous subpart.

Seeing how Indiana figures out the path to the skull, the ancient powers of the skull start to confuse Indiana.

Now, when Indiana takes an action, **the action is changed to a different action** according to a particular probability distribution denoted $p(a'|s,a)$. Given state s and action a, the action is changed to a' with probability $p(a'|s,a)$. To confuse him further, Indiana will receive reward as if his action did not change at all.

Indiana tries to adopt those changes to Q-value iteration.

$$\text{Regular MDP:} \quad Q(s,a) \leftarrow \sum_{s'} T(s,a,s')\big[R(s,a,s') + \gamma \max_{a'} Q(s',a')\big]$$

$$\text{New formulation:} \quad Q(s,a) \leftarrow \textbf{(i) (ii) (iii) (iv)}\Big[\textbf{(v)} + \textbf{(vi)}\Big]$$

(i)	○ 1	○ $\sum_{a'}$	○ $argmax_a$
(ii)	○ $p(a'\|s,a)$	○ $T(s,a,s')$	○ $\sum_{s'}$
(iii)	○ $T(s,a',s')$	○ $\sum_{s'}$	○ 1
(iv)	○ $T(s,a',s')$	○ $p(a'\|s,a)$	○ γ
(v)	○ $R(s,a',s')$	○ γ	○ $R(s,a,s')$
(vi)	○ $\gamma * max_{a''}Q(s',a'')$	○ $\gamma * Q(s',a')$	○ $\gamma * argmax_a Q(s',a)$

(c) [2 pts] The rest of this question is independent of the previous subpart.

Indiana's enemies are not sleeping and they also are trying to get the skull power. They are also trying Q-value iteration. They heard old stories about how people approaching the lake lose control of their movement.

Consider a modified MDP where the agent can choose any action a, but the next state s' will be determined regardless of the action. In other words, $T(s,a,s')$ can be changed into $T(s,s')$, but $R(s,a,s')$ stays the same. Looking at the policy extraction step, Indiana's enemies noted that to get the policy, they do not even need any Q-values or V-values.

Write an expression for deriving the optimal policy **without using any Q-values or V-values**.

$$\text{Regular MDP:} \quad \pi(s) \leftarrow argmax_a \sum_{s'} T(s,a,s')\Big[R(s,a,s') + \gamma * max_{a'} Q(s',a')\Big]$$

Q7. [12 pts] Reinforcement Learning

In this class, we have studied several reinforcement learning (RL) methods. In this problem, we will examine properties of several extensions of these approaches.

(a) [4 pts] Select all true statements about Q-learning.

For this subpart, assume that during Q-learning, we visit every state-action pair an infinite number of times.

□ At convergence, Q-values obtained from Q-learning will lead to the optimal policy of the MDP, π^*.

□ At convergence, Q-learning always finds a deterministic policy.

□ Q-learning cannot converge in any MDP when $\gamma = 1$.

□ Assume that you derive a deterministic policy from a Q-function Q. The policy is denoted by $\pi(s) = \arg\max_a Q(s, a)$. Then, the learned Q-values $Q(s, a)$ for $a \neq \pi(s)$ represent the expected discounted reward of a valid policy in the MDP.

□ When running policy iteration, the $Q(s, a)$ values in the table of Q-values at intermediate iterations represent the correct Q-function for some policy in the MDP.

○ None of the above

(b) In this subpart, we will consider how standard Q-learning will behave in two different types of MDPs:

- Infinite-horizon MDP **with terminals**: there are certain states, denoted as $\mathcal{S}_T$, such that if the agent reaches these states, it terminates and gets no reward.
- Infinite-horizon MDP, **without terminals**: there are no states where the MDP terminates.

Formally, for any state s in the MDP, we define a function $\tau(s) \in \{0, 1\}$. In other words, $\tau(s) = 0$ or $\tau(s) = 1$ for all s. $\tau(s) = 1$ implies that $s \in \mathcal{S}_T$ (i.e. s is a terminal state).

(i) [2 pts] Fill in the blank for the Q-learning update with terminations, in terms of $\tau(s)$.

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha \left[R(s, a) + \gamma \boxed{} \max_{a'} Q(s', a') \right]$$

(ii) [4 pts] Consider two MDPs, $\mathcal{M}_1$ and $\mathcal{M}_2$, which are identical, except that their reward functions are shifted versions of each other:

- For every state-action pair (s, a) in $\mathcal{M}_1$, $0 \leq R_1(s, a) \leq 1$
- For every state-action pair (s, a) in $\mathcal{M}_2$, $R_2(s, a) = R_1(s, a) - 2.0$

Assume there are no cycles in the MDP (i.e. you cannot keep accumulating reward infinitely).

True or false: The policies found by Q-learning on $\mathcal{M}_1$ and $\mathcal{M}_2$ **without terminals** are **not** identical.

○ True, because reward functions are not the same in the two MDPs.
○ True, because Q-learning would not accumulate negative reward.
○ True, but not for the reasons above.
○ False, shifting a reward function does not affect the optimal policy.
○ False, because the reward function is now negative.
○ False, but not for the reasons above.

True or false: The policies found by Q-learning on $\mathcal{M}_1$ and $\mathcal{M}_2$ **with terminals** are **not** identical.

○ True, because reward functions are not the same in the two MDPs.
○ True, because there are MDPs where we might terminate instead of attaining negative reward.
○ True, but not for the reasons above.
○ False, shifting a reward function does not affect the optimal policy.
○ False, because the reward function is now negative.
○ False, but not for the reasons above.

(c) [2 pts] Consider this modification to the Q-learning update:

$$Q(s, a) \leftarrow \alpha \cdot \left(R(s, a, s') + \gamma \frac{\sum_{a'} Q(s', a')}{\|a'\|} \right) + (1 - \alpha) \cdot Q(s, a)$$

where $\|a'\|$ is the number of actions available from state s'.

If we run Q-learning with this update on an MDP, the learned policy is obtained by: $\pi'(s) = \max_a Q(s, a)$.

Would $\pi'(s)$ be the optimal policy?

○ Yes, but only if the MDP has deterministic transitions.
○ Yes, but only if the MDP does not have cycles.
○ Yes, but only if we visit every state-action pair infinitely often, and we decrease the learning rate appropriately.
○ No, $\pi'(s)$ will be the least optimal policy.
○ No, $\pi'(s)$ will be an arbitrary policy that may be random, optimal or sub-optimal.
○ No, $\pi'(s)$ will be a greedy policy that always maximizes the immediate next reward (ignoring future rewards).