UC Berkeley – Computer Science
CS188: Introduction to Artificial Intelligence
Josh Hug and Adam Janin

Midterm I, Fall 2016

This test has **8** questions worth a total of **100** points, to be completed in 110 minutes. The exam is closed book, except that you are allowed to use a single two-sided hand written cheat sheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided.

Write the statement out below in the blank provided and sign. You may do this before the exam begins. Any plagiarism, no matter how minor, will result in an F.

"I have neither given nor received any assistance in the taking of this exam."

____

____

Signature: ______________________________

Name: ______________________________ **Your EdX Login: ______________________________**
SID: ______________________________ **Name of person to left: ________________________**
Exam Room: ________________________ **Name of person to right: ______________________**
Primary TA: ________________________

- ◯ indicates that only one circle should be filled in.
- ▢ indicates that more than one box may be filled in.
- Be sure to fill in the ◯ and ▢ boxes completely and erase fully if you change your answer.
- There may be partial credit for incomplete answers. Write as much of the solution as you can, but bear in mind that we may deduct points if your answers are much more complicated than necessary.
- There are a lot of problems on this exam. Work through the ones with which you are comfortable first. **Do not get overly captivated by interesting problems or complex corner cases you're not sure about.** Fun can come after the exam: There will be time after the exam is try them again.
- Not all information provided in a problem may be useful.
- Write the last four digits of your SID on each page in case pages get shuffled during scanning.

Problem	1	2	3	4	5	6	7	8
Points	11.5	9.5	14.5	12	8.5	16	19	9

Optional: Mark along the line to show your feelings on the spectrum between :(and ☺.

Before exam: [:(____________________ ☺]
After exam: [:(____________________ ☺]

1. Munching (11.5 pts)

i) (7.5 pts) Suppose Pac-Man is given a map of a maze, which shows the following: it is an ***M*-by-*N*** grid with a single entrance, and there are **W** walls in various locations in the maze that block movement between empty spaces. Pac-Man also knows that there are **K** pellets in the maze, but the pellets are not on the map, i.e. **Pac-Man doesn't know their locations in advance**. Pac-Man wants to model this problem as a search problem. Pac-Man's goal is to plan a path, before entering the maze, that lets him eat the hidden pellets as quickly as possible. **Pac-Man can only tell he has eaten a pellet when he moves on top of one.**

Which of the following features should be included in a minimal correct state space representation? For each feature, if it should be included in the search state, calculate the size of this feature (the number of values it can take on). If it should **not** be included, **very briefly** justify why not.

Feature	Include?	Size (if included) or Justification (if not included)
Current location	◯ Yes / ◯ No	
Dimensions of maze	◯ Yes / ◯ No	
Locations of walls	◯ Yes / ◯ No	
Places explored	◯ Yes / ◯ No	
Number of pellets found	◯ Yes / ◯ No	
Time elapsed	◯ Yes / ◯ No	

ii) (1 pt) Which approach is best suited for solving the K hidden pellets problem? Completely fill in the circle next to one of the following:

◯ **State Space Search** ◯ **CSP** ◯ **Minimax** ◯ **Expectimax** ◯ **MDP** ◯ **RL**

iii) (2 pts) Now assume that there are K_R red pellets, K_G green pellets, and K_B blue pellets, that Pac-Man **knows the locations and colors of all the pellets**, and that Pac-Man wants **the shortest path that lets him eat at least one pellet of each color,** i.e. one red, one green, and one blue. What is the total size of the search state space, assuming a minimal correct state space representation?

iv) (1 pt) Which approach is best suited for solving the one-pellet-of-each-color problem? Completely fill in the circle next to one of the following:

◯ **State Space Search** ◯ **CSP** ◯ **Minimax** ◯ **Expectimax** ◯ **MDP** ◯ **RL**

2. Uninformed Search (9.5 pts)

Part A: Depth Limited Search (DLS) is a variant of depth first search where nodes at a given depth L are treated as if they have no successors. We call L the **depth limit.** Assume all edges have uniform weight.

i) (1 pt) Briefly describe a major advantage of depth limited search over a simple depth first tree search that explores the entire search tree.

ii) (1 pt) Briefly describe a major disadvantage of depth limited search compared to a simple depth first tree search that explores the entire search tree.

iii) (2 pts) Is DLS optimal? ◯ Yes / ◯ No   Is DLS complete? ◯ Yes / ◯ No

Part B: IDDFS. The iterative deepening depth first search (IDDFS) algorithm repeatedly applies depth limited search with increasing depth limits L: **First it performs DLS with L=1, then L=2, and so forth, until the goal is found.** IDDFS is neither BFS nor DFS, but rather an entirely new search algorithm. Suppose we are running IDDFS on a tree with branching factor B, and **the tree has a goal node located at depth D**. Assume all edges have uniform weight.

i) (2 pts) Is IDDFS optimal? ◯ Yes / ◯ No   Is IDDFS complete? ◯ Yes / ◯ No

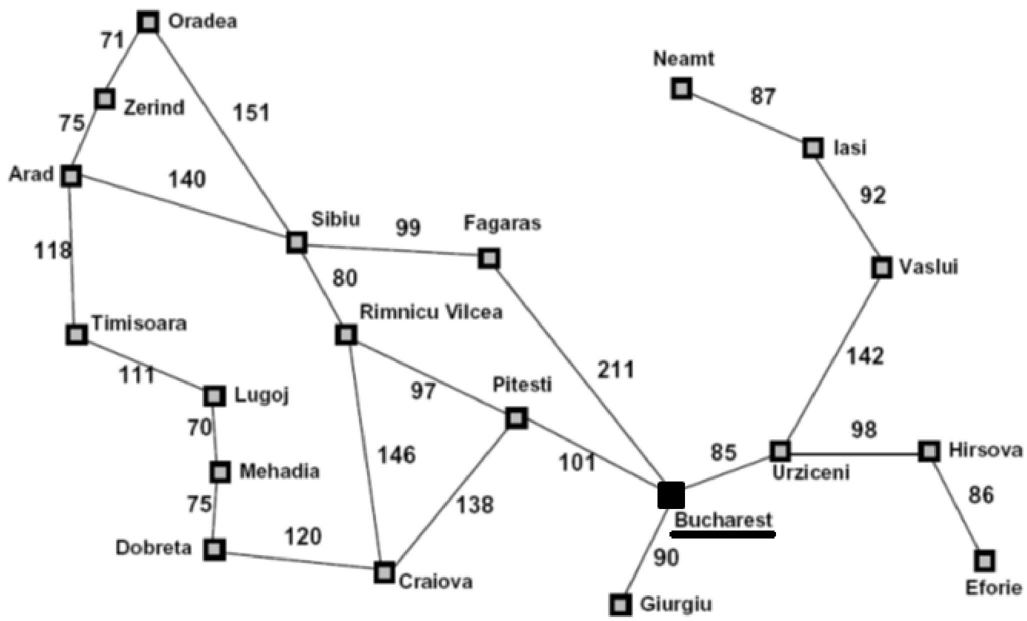
ii) (1 pt) In the worst case asymptotically, which algorithm will **use more memory** looking for the goal?

◯ IDDFS / ◯ BFS

iii) (2.5 pts) In project 2, you used DLS to implement the minimax algorithm to play Pac-Man. Suppose we are instead writing a minimax AI to play in a speed chess tournament. Unlike the Pac-Man project, there are many pieces that you or your opponent might move, and they may move in more complicated ways than Pac-Man. Furthermore, you are given only 60 seconds to select a move, otherwise you are given a random move. **Explain why** we'd prefer to implement minimax using IDDFS instead of DLS for this problem, and **explain how** would we need to modify IDDFS (as described above) so that it works for this problem.

3. Aliens in Romania (14.5 pts)

Here's our old friend, a road map of Romania. The **edges** represent **travel time along a road**. Our goal in this problem is to find the shortest path from some arbitrary city to Bucharest (underlined in black). You may assume that the heuristic at Bucharest is always 0. In this question, **L(X, Y) represents the travel time if we went in a straight line from city X to city Y**, ignoring roads, teleporters, wind, birds, etc. You should make no other assumptions about the values of L(X, Y).



Part A: Warm up.

i) (6 pts) For each of the heuristics below, completely fill in the circle next to "Yes" or "No" to indicate whether the heuristic is admissible and/or consistent for **A* search**. You should fill in 6 answers (one has been provided for you). **Reminder, h(Bucharest) is always zero.**

Heuristic h(C) = [except C = Bucharest]	**Admissible**	**Consistent**
0	◯ Yes / ◯ No	◯ Yes / ◯ No
90	◯ Yes / ◯ No	◯ Yes / ◯ No
L(C, Bucharest)	◯ Yes / ◯ No	◯ Yes / ◯ No
min(L(C, Bucharest), 146)	◯ Yes / ◯ No	

ii) (2.5 pts) For what values of x is h(C) = max(L(C, Bucharest), x) guaranteed to be admissible? Your answer should be in the form of an expression on x (e.g. "x = 0 OR x > 102").

Part B: Teleporters

Aliens build a teleporter from Oradea to Vaslui. This means that we can now get from Oradea (top left) to Vaslui (near the top right) with zero cost.

i) (1 pt) Update the drawing on the **previous page** so that your graph now reflects the existence of the teleporter.

ii) (5 pts) Completely fill in the circle next to "Yes" or "No" to indicate which of the following are guaranteed admissible for A* search, given the existence of the alien teleporter. **Reminder, h(Bucharest) is always zero, and L(A, B) is the travel time between A and B assuming a straight line between A and B ("as a bird flies").**

Heuristic h(C) = [except C = Bucharest]	**Admissible**
77	◯ Yes / ◯ No
L(C, Bucharest)	◯ Yes / ◯ No
max(L(C, Bucharest), 80)	◯ Yes / ◯ No
max(L(C, Bucharest), L(C, Oradea))	◯ Yes / ◯ No
min(L(C, Bucharest), L(C, Oradea))	◯ Yes / ◯ No

Survey Completion

Enter the secret code for survey completion: ______________________________

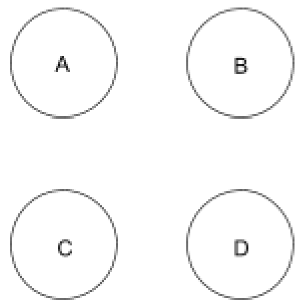
If you took the survey but don't have the secret code, explain: ____

4. CSPs (12 pts)

In a combined 3rd-5th grade class, students can be 8, 9, 10, or 11 years old. We are trying to solve for the ages of Ann, Bob, Claire, and Doug. Consider the following constraints:

- No student is older in years than **C**laire (but may be the same age).
- **B**ob is two years older than **A**nn.
- **B**ob is younger in years than **D**oug.

The figure below shows these four students schematically ("A" for **A**nn, "B" for **B**ob, etc).



i) (2 pts) In the figure, draw the arcs that represent the binary constraints described in the problem.

ii) (1 pt) Suppose we're using the AC-3 algorithm for arc consistency. How many total arcs will be enqueued when the algorithm begins execution? Completely fill in the circle next to one of the following:

◯ **0** / ◯ **5** / ◯ **6** / ◯ **8** / ◯ **10** / ◯ **12**

iii) Assuming all ages {8, 9, 10, or 11} are possible for each student before running arc consistency, manually run arc consistency on **only** the arc *from* A *to* B.

A. (2 pts) What **values on A** remain viable after this operation? Fill in all that apply.
☐ 8 / ☐ 9 / ☐ 10 / ☐ 11

B. (2 pts) What **values on B** remain viable after this operation? Fill in all that apply.
☐ 8 / ☐ 9 / ☐ 10 / ☐ 11

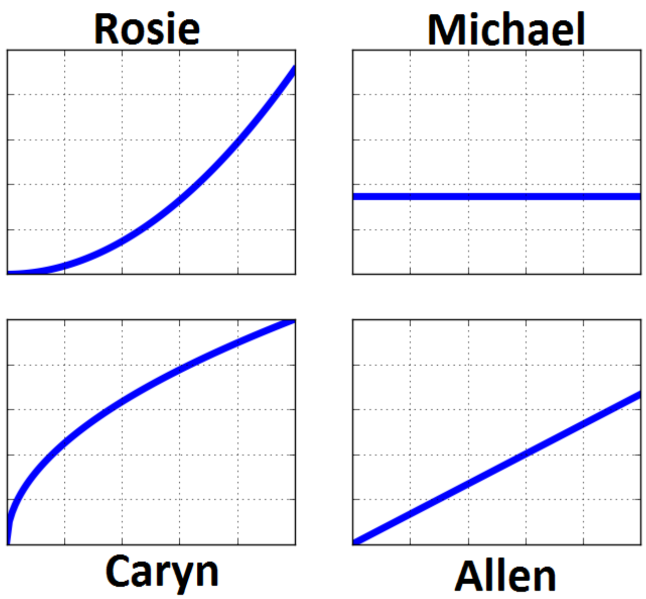
C. (1 pt) Assuming there were no arcs left in the list of arcs to be processed, which arc(s) would be added to the queue for processing after this operation?

iv) (4 pts) Suppose we enforce arc consistency on all arcs. What ages remain in each person's domain?

Ann: ________ **B**ob: ________ **C**laire: _______ **D**oug: _______

5. Utilities (8.5 pts)

Part A: Interpreting Utility. (3.5 pts**)** Suppose Rosie, Michael, Caryn, and Allen are four people with the utility functions for marshmallows as described below (e.g. Michael doesn't care how many marshmallows he gets). The x-axis is the number of marshmallows, and the y-axis is the utility. Rank our four TAs from **least risk-seeking (at the bottom)** to most risk-seeking (at the top) using the four lines provided. **If two people have the same degree of risk-seeking, put them on the same line**. You may not need all four blanks (e.g. if two people are on the same line).



Rank in terms of risk-seeking behavior:

Most: ________________________

____

____

Least: ________________________

Part B: Of Two Minds. (5 pts) Some utility functions are sometimes risk-seeking, and sometimes risk-averse. Show that $U(x) = x^3$ is one such function, by providing two lotteries: one for which it is risk-seeking, and one for which it is risk-averse. Provide justification by filling out the table below. Averse is just the opposite of "seeking". Specifically, it means "having a strong dislike for something," in this case, risk.

	Risk-seeking:	Risk-averse:
Lottery		
Utility of lottery		
Utility of expected value of lottery		

6. Games / Mini-Max: The Potato Game (16 pts)

Aldo and Becca are playing the game below, where the left value of each node is the number of potatoes that Aldo gets, and the right value of each node is the number of potatoes that Becca gets (i.e. Aldo: left, Becca: right). Unlike prior scenarios where having more potatoes results in more utility, Becca and Aldo will have a more complex view of what makes a "good" distribution of potatoes.

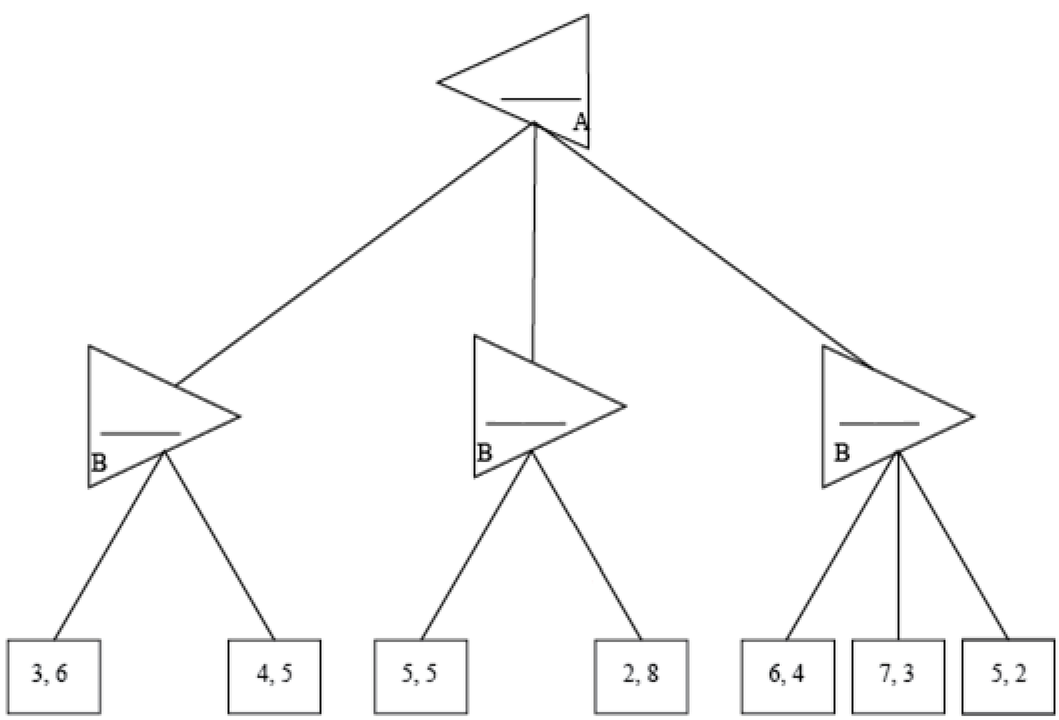
Becca will use the following thought process to decide which move to take:

- Among all choices where she gets at least as many potatoes as Aldo, she'll pick the one that maximizes Aldo's number of potatoes (very nice!).
- If there are no choices where she gets at least as many potatoes as Aldo, she'll simply maximize her own potato count (ignoring Aldo's value).

Aldo will do the same thing, but substituting "he" for "she", "her" for "his", and "Becca" for "Aldo". The rules above effectively tell us how Becca (and Aldo) rank the utility of any set of choices.

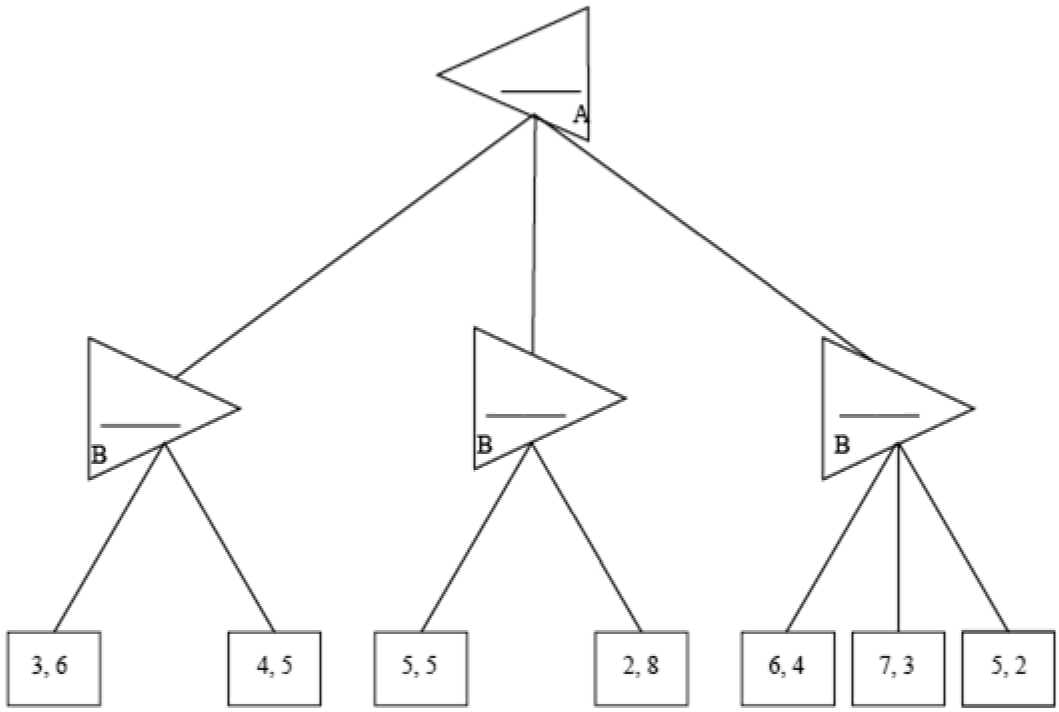
i) (4 pts) Fill in the blanks below with the choice that each player would make at each stage. Assume that both Aldo and Becca are trying to maximize their own utility as described above.

ii) (3 pts) Assume that Aldo and Becca know that **the sum at any leaf node is no more than 10**. Cross out the edges to any nodes that can be pruned (if none can be pruned then write "no pruning possible" below the tree).



iii) (3 pts) Describe a general rule for pruning edges, assuming that the sum at any leaf node is no more than 10.

iv) (4 pts) Suppose that **Becca attempts to minimize Aldo's utility instead of maximizing her own utility, and that Aldo knows this, and tries to maximize his own utility -- taking into account Becca's strategy**. Repeat part ii. There is no need to cross out edges that are pruned.



v) (2 pts) Suppose that **Becca chooses uniformly randomly and Aldo knows this and tries to maximize his own utility -- taking into account Becca's strategy**. Which move will Aldo make to maximize his expected utility, assuming he treats Becca as a chance node: left, middle, or right? If there is not enough information, pick "Not Enough Information".

◯ Left ◯ Middle ◯ Right ◯ Not Enough Information

7. The Spice Must Flow (19 pts)

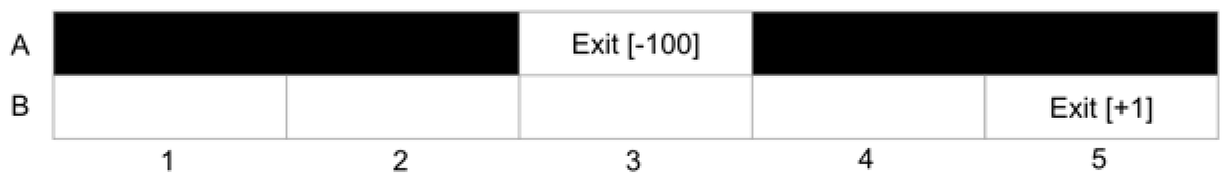
i) (3 pts) Suppose we have a robot that has 5 possible actions: {↑, ↓, ←, →, Exit}. If the robot picks a direction, it has an **80%** chance of going in that direction, a **10%** chance of going 90 degrees counterclockwise of the direction it picked (e.g. picked →, but actually goes ↑), and a **10%** chance of going 90 degrees clockwise of the direction it picks.

If the robot hits a wall, it does not move this timestep, but time still elapses. For example, if the robot picks ↓ in state B1 and is 'successful' (80% chance), it will hit the wall and not move this timestep. As another example, if it picks → in state B1, but gets the 10% chance of going down, it will hit the wall and not move this timestep

In states B1, B2, B3, and B4, the set of possible actions is {↑, ↓, ←, →}. In states A3 and B5, the only possible action is {Exit}. Exit is always successful. The blackened areas (A1, A2, A4, and A5) represent walls.

All transition rewards are 0, except R(A3, Exit) = -100, R(B5, Exit) = 1.

In the boxes below, draw an optimal policy for states B1, B2, B3, and B4, if there is no discounting, i.e. $\gamma = 1$. Use the symbols {↑, ↓, ←, →}.



ii) (2 pts) Give the expected utility for states B1, B2, B3, and B4 under the policy you gave above
.

V*(B1): _______ V*(B2): _______ V*(B3): _______ V*(B4): _______

iii) (2.5 pts)What is V*(B4), the expected utility of state B4, if we have a discount $0 < \gamma < 1$ and use the optimal policy? Give your answer in terms of γ.

V*(B4): ________________________

Part B: Spice. Now suppose we add a sixth special action **SPICE**. This action has a special property that it always takes the robot **back to its current state**, with a reward of 0. This action might seem useless, except for one thing - the robot will always get its top preference on its next action (instead of being subject to noise).

For example, suppose that the robot is in state **s1** and picks action **a** which has a chance **p2** of moving to **s2**, and a chance **p3** of moving to **s3**. If the robot takes the **SPICE** action at timestep t, and **a** at timestep $t+1$, the robot will not end up randomly in **s2** or **s3**, but will instead go to its **preference** of **s2** or **s3**. It will still collect R(**s1**, **a**, **s'**), which is unchanged. Preferring **s2** or **s3** does not take an additional time step, e.g. if the robot is in B1 at timestep 0, chooses SPICE, then chooses ↓ but prefers the outcome B2, the robot arrives in B2 at timestep 2.

The powers granted by **SPICE** last one timestep. **SPICE** may be used any number of times.

i) (2 pts) Give an optimal policy for B1, B2, B3, and B4, assuming no discounting, i.e. $\gamma = 1$. **On the left of each slash,** write the optimal action if the previous action was not **SPICE**, and on the **right side of each slash**, write the optimal action if the previous action was **SPICE, assuming the robot always "prefers" the outcome that points in the same direction as its action (e.g. if it picks →, it prefers going right)**. In total, you should write 8 actions. Use the symbols {↑, ↓, ←, →, S}, where "S" represents indicate the spice action.

	1	2	3	4	5
A			Exit [-100]		
B	/	/	/	/	Exit [+1]

ii) (2.5 pts) Assuming we have a discount $0 < \gamma < 1$, what is V*(B4) if the previous action was not **SPICE**? Give your answer in terms of γ. It is OK to leave your answer in terms of a max operation.

V*(B4): ____

OFFICIAL RELAXATION SPACE. Draw or write anything here:

Part C: Bellman Equations. (You can do this problem even if you didn't do B, but it'll be harder)

In class, we derived the Bellman Equations for an MDP given below.

$$Q^*(s,a) = \sum_{s' \in S_{(s,a)}} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

$$V^*(s) = \max_{a \in A_s} Q^*(s,a)$$

You'll notice these look slightly different than the version on your cheat sheet, but these equations represent exactly the same idea. The only difference is that to improve the clarity of these equations, we've specified the sets over which we are summing and maximizing. Specifically, $S_{(s,\ a)}$ is the set of states that might result from the action (s, a), and A_s is the set of actions that one can take in state s. **For this problem, assume that A_s does not include the special SPICE action.**

i) (3.5 pts) Derive Q*(s, SPICE). Assume that the previous action was not **SPICE**. Your answer should have two max operators in it. Hint: Consider drawing the diagram we used to derive the Bellman Equation.

ii) (3.5 pts) Derive V*(s), the value of a state in this MDP. You should account for the the fact that the **SPICE** action is available to the robot. Assume that the previous action was not **SPICE**.

8. The Reinforcement of Brotherly Love (9 pts)

Consider two brother ghosts playing a game in an M-by-N grid world, Ghost Y is the Younger brother of Ghost O. Each ghost takes 100 steps, trying to pet cats which randomly appear in the maze. The reward for the game is equal to the number of cats petted. Each ghost has a policy for how to run through the maze. Ghost O being the older brother has a policy that has a **higher expected cumulative reward** (i.e. $V_Y(s) \leq V_O(s)\ \forall s \in S$). However, Ghost O is **not necessarily optimal** (i.e. $V_O(s) \leq V^*(s)\ \forall s \in S$). Ghost Y wants to learn from Ghost O instead of starting from scratch.

Part A: Q-Learning

We will now explore how Ghost Y can achieve this, while performing **Q-Learning**. Ghost Y starts out with a Q function **initialized randomly** and wants to catch up to Ghost O's performance. Denote the Q functions for each Ghost as Q_Y, Q_O. In order for Ghost Y to converge to the optimal policy, he must properly balance **exploring** and **exploiting**. Consider the following exploration strategy: At state s with probability 1-ϵ choose $max_a\, Q_Y(s, a)$ otherwise choose $max_a\, Q_O(s, a)$.

(2 pts) Is this guaranteed to converge to the optimal value function V*? Briefly justify.

◯ Yes / ◯ No ____

Part B: Model-Based RL

Ghost Y decides model free learning is scary, and decides to try to learn a model instead. Ghost Y will watch Ghost O try to catch Pac-Man and estimate the **transition** probabilities and **rewards** from Ghost O's demonstrations (i.e. learn a model for T(s,a,s') and R(s,a,s')).

i) (2 pts) For Ghost Y to learn the correct and exact values of R(s,a,s') for all state/action/next_state transitions, what must be true about the episodes he watches Ghost O perform?

ii) (2 pts) For Ghost Y to learn the correct and exact values of T(s,a,s') for all state/action/next_state transitions, what must be true about the episodes he watches Ghost O perform?

iii) (2 pts) Assuming T(s,a,s') and R(s,a,s') are learned **exactly**, what algorithm/s below could be helpful for Ghost Y to decide which actions to take from each state, based **only** on information in his learned model?

☐ **Value Iteration** ☐ **Policy Iteration** ☐ **Q-Learning** ☐ **TD-Learning**

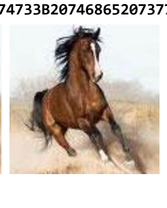
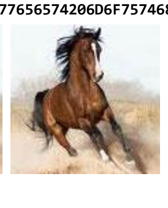
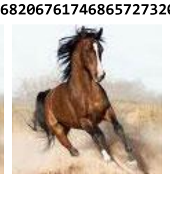
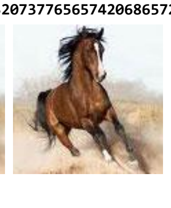
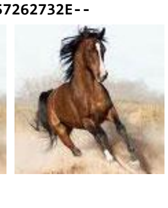
iv) (1 pt) Let π be any of the policies learned in part iii. Does following π result in maximized expected utility for every state, i.e. is $V^{\pi}(s) = V^*(s)$ for all states? [Consider all policies]

◯ **Yes** ◯ **No** ◯ **Not enough information**

DO NOT WRITE ON THIS PAGE. IT WILL NOT BE GRADED.

504B03041400090008005C694549D992860B96110000CA1100000C001C00676C7970686167652E706E675554090003505EF557FC5EF55775780B000104F501000004140000007A41EA7E54C5085
ADCCA44BADCC524A15475544046F748AE7372C6D2CD4E47E8B7E7CB6596470370A4BEBE144CEB2EE0E3E5DFFF994D70C833845C625F91596A82636CD38E61C369058A3AB8B54968A81ADC716BC6
04AFB759A0E81DB8405B66B022B0FE543490278C3891DDDF4EBE49697FF9775EF828ECA9457A271D0E0209516E510E4CEEFD45DF58789D932A1BC43EC722D9D24048AF0F15DBDE9539AC905C39F
65ACFE3C6EF24062195C174FAC0A8AD794ADB602124B06415F981E0EC7053722D3C75101B9B30BA9A42A6699B8C3B141934B53346145309A8BEE3381E0957A594E8AE6F66326B56D2BFF0E85DA3
DA741CB2EDFD18F36026C91BFAF65E3BFFFCF1A40E9CB6901CAA6688A2870182E03562037B144FFE2A611C1BBAE646BCBFF26DC95FD8C4AF85B499C36084074E6906BB08DE2BDB85CAC29B96AE3
B7BD136584C50E1C74631E7EFD4E2226E0A13D9024F76F8ED697B13840232198E4B6C00B12AED6E3D103081BEDAD890BA45B6EA73B8045E872387BEBB4ABF8B7826E8F9E26B12B086E52C192789
370E8666214D446C079A8FA1B0FD65E381F995F7BCF5A004016C0542C83B027959D74D19827A4C0CC8B499A49540D4A942ADA2196766C87AF7E343AA8DA1FDDEDF048C757532FAF42F7BC0AA380
D8CF8DAC9C850D1609C4AF36EAD8570F38C9F990E187B3FFF384CA8151A3E112C458736AAFF65291AD73B998F659419E4181E6190203A2BDC0DE0F6A03045EC793DAC9F495675F3B3A74A9C8D05
B466725BA3C73A6633E5DCE90D18EC1FF8DBA7FC2E9B501972CA674CB6C6ECB981EE88C96CCEA40E77D1966FB0252F8E4D6F7F6DEACA986C3338F097A36654ED06BAC4D4ECE1DD79DDFC0F7AA5C
52D129F9D21FBB082CBA1D1CC89A891D91B15BDA4BD8014058A7618A8C31A0A2BDD63C6E3A4D229B90F898168746690FE533C0B0505EA19A49D2F73A48B2B8E85E4AFD1FB6B20F131E130C9A793
6FBA7A35111D6ED6F7309DC36C88DA4052D544A12A040FF96D946DB9471DBFDEBC736BA0F29BA703DF982BFAABBAAA001A9580564FC3764F89473D6D6A2381E0C57DD0DAE6298AEDD0C75D81EFD
81C94C9EFC2AA7EA5900D85EBB706B51B1ACCD1949C18B4180BA3DE65FB9B6EC8D7F7533C02A18779FA6BBF8E0F386E6DC00CC046DDE29E72839537E5106862DCBB57056AE8F8207CC855C65C16
86FD335963F23928068388A9BE75E3C69B26E64A689C8B00E505CFA80243FA36B26B773346974678B4A8D7C99BEBF75E061D840B1137FA1967C1656C3DBFDF04C82D65D01513A8B5BED93B8F760
75F6A8B979F4D1061AB0A64EE808F39819EDFD476B0124A55C892C8F14FB5CA197E4079578FB0397B6992134A637045A8C9EB8238F02CE0ECA834F194FCC0814EF9495D16F70D5B2B7AFEF9199B
9CA2FEF2A60E8913A247088345A6983C7ACDF01358CB5C4B5BB093380A48D788D24C1DBA0DDCF207C697A9454F064A16E28513E2DB92B0D0FDA62F221B86BB92F061F791CFE60FC1EA7A46DB9C5
FA3A8063A265D80A035B37EDC859D40813BF1E1068488DA4B925088821713A979776B722A0ED17604A92E2D053B1A798C5F4F9AD08038153C6DDAAE3A31772EEF22249F30044C8450DC7D19513D
C4B897CA46C21A09FA13E7EFF01CF2D36EE3DD4474A3C460B4FE000A74DA090C1A31A9F768E0EDCAD2C967E6F1648D69D3D493ED3A68C0B9C46450880DB77CDD9ED6A82645FD9982AD419B1BAD0
725FA2E34936A7A05AD6B3E55D88D4DF41E50103BAB4F282620D66D6C06B7EB14465B5C56F76CCAF2E07B9B052A57CFED32AD203E71F85267C3F40122E3EBA65080D26BBDEEB72D1C001528F682
E1B6D6ACC9544758C99B42DDBCF7D58926E2DAD3975942C4E48DCBB7057454D305FB7A8A7197C33152A0F0C3AE7DA62D2ECEA6C9ACA34048DCA071AA7254AE67F3091048CCF9926E44D8C4A48D9
C081A37D52FB7BFB8AF1C415A99CE4FEF4133CE5DC6AE310A370CFE377F4FB63E3617FBE58C03E74A8ECE66C84EADBE87ACD500039460FD2DD02765C9E41DBC89D8F94E808F973A9CF33E6C439C
1DF00FCA297F2E02233A89113CEE671B2D77B988BC82266836CD6329232546E805DBB780DE19DECAE05AB6A874E897B4A61EF5EC608B7B1D8FD6BCBCB7D44A0E7E62B727D2EBD307813DEF94EA9
9CE0EA74A884A8239893DD85F94DB3664C2B035FFE95E68B36303E32DBB2D16D6F9A2ADC66B939541EB2E1DBAC5E61A7B00BDFBF78818FFABE1BEBE4D8B6850618DD8AE9083C0C7C34E3D59F999
538E69E1CA1A0B482404909F9732344E141658C82AC4535D3E553C7798216F6A415B902B420D50B3CA66F2AAD495811D9442936D0922DE67C18AA571AE2D70A683380CF41724F1C8E4DC20937CB
67D9A06E8CAEF2A62ADD8BE97339F288A1B386A9F1F631315155C289F5734C4D785353B33AE7F37F109BDD05B80373E6FDE8F78DB4E9B087E69E44E5801F2F64BA38926C8FBF88BFB4BE2C4370C
A938FD3921F51BF7EA911186DC37AFF11D1819698F9203FC46AEA5BAF54B775642352E6535B2D7ECA81D21DB9FEA9A1C624297B5B762D68BB5D05F91CAE2C1D0FDE59246911C4DABE3E13B1E933
A8CBFB57DD6EB8F84A03F0999E6750A1581547BB7A6E01970B4DC30665C165E4C2BF5D1C3E9F95571CB6EA77F7D1E47C3B6D80A28999F6F1D56F3F7CEF7F1FCC0033EFECA476DAD30265804181A
6426C49F0A4420C9D73B3124318D0BDB52D64B79A73ECBCCE996C5482D9C36430C2A2DD0E38F0C4B2242D280D3B735A159D5B1624F9901481235E2F0A7AB11B92BF498B71E1C605A3D4A71C950F
5FD4463181FA1CECE91BF921B017E347EB47749FC41EE0E4387D4619CE6966D22E0B6E0F9F4156FF88EB0A0E85C4EC8703F3EB3731236C842A609E43F3F80A74CE9441634F5F51226E126D4B7FB
C431A710C5D44F87DE66901EA7BECE03D718D9A52CD0DA3CA3F1F3C65937FCF6031A0690B5BA994F0408D54A1017B5202D5E700E3A23F970703A94E01EB67D79DFDCFB637D46D1BB5BD021C65F3
819D6C1C5CF4D8DA7CD093153FC79E52A2B2992634F86BE54D5DA4C2F5E0A34819BD2EAC44402FC13404DAB1A79E053386DC4D1B028FD36DC432A4D038DB70621B5C644DA7256213EB2E8883D1B
92D8B2324FA13725EC3738839DE4C6A7243A142EA129DBAFD5995E29A3FB3D1220236AF15AA85A5B47FDB8CE66EDCE53A21905DBBB0DF5E5D26B7474BF2B4B47E67619B02ED5DC83B801892034A
DD285714FE5F329C48C9538918FE3193E18F8E726C462D4C69A4102E90BC8CB699EC95555EA2AFDDE601136EA36BC6B0E00540438DE8D11F771BA01E6DB501E4D2B5939FEF803C0305F8976C5DD
1FF7EB77BF09A6779ACA0C31A18A431176E7178875B94F5D6FFF71AAF94B09F3A709F00A3E6B9E45A3CC1F217E9CA488F85985F5182C2700BEB65FFB5F4AFB7E123BC73A280E9AF18E405FFB438
9667E87D4695B2E068FA8AD979108A59518D022913F8AE2B22EB3470A132FDF006B5F35BA23E6B9106517172090F22397FBCF2736AAD75DD2E5B0578D7C344C175B516325654DF4F54D49F82604
0BD3A692A352D41CF191689A54CCB3C941779BEFBE38AE04ADF395D3EB59A62074887F1BDFC705D15148A10735C2C2EB5084B22551DEB3593BC5739BF4BE66EE985D1915FAB316B612CDBAC862C
24F9E644B6DEF0AC266D24D63C82AFD798AB6C644BF45740B7E5A1CFBAD946A1B5ACF961CE2ECB44A31DB0A30D54A1492C348AD939890C280E2E4D6C4DE46FBAC1D6CA2B31421E60A36F14E891A
0397EA353CA765A9F947BF0CE8BAA4EFE757BDF85AAD28FE061504C02E2C9ECC7D2CD1BF2ADB2DD6F687477053038182B19312ECF85E907F527AE7131E1E09DB634F9F594307FC4CC91414D1D96
B4A73F62550952958B3DB377BF6D9C34BA7ACAD17CDCE1A493D24011B9AFEC2D232D1B41562282DFBE2A942AC34BB9837FABDBD09132AB781D1810730995E28E8E5C2058B01E00B9726A920FDFB
BE331E81E6BCAD5DE42E314407F0579A1D2E01870B2FF195EF036DD839BFCC05618E935E40EC0AEBE137B164148BF48C673C46DA112A333CE3A153D957CA1DD0082E832B22E5AA4E92CD6C901E8
C25BF8D32A7A953CE7493EA0003D8B2686CBF7DBDE866BC48BAD103DE86B22034CA17E2FA812ED6D79C7B40808D928A8F106C8CCD7B8D410359127139F149720E737D12C7C270E7A645519E6DE1
1912F55F523699C5417319E4B519AC93121F49B617F6CBD8EFC3A092A7A5916CB1AB66F1B31EBC3EE09D42DBB45B687CF6B55C2F49F19A11431D537581D9FFFF240D1DF68EBB27DAA1A54BDA148
EABC0FFFE401C10D63F665F3F44DD079AFBEA69A6A1F152A4745EF4C2B49E184D6D9B2F38097B4B971DCA89461BBA5690B3618DA28F77BC6ADB679A9E8AB03A4840B614A0F683BAA4DFD242E98A
91F87612218D16149DE4AA5406C531C4F3510750EFD9EB351558AE894495FF32E82D942C5762BD74AFCEFA489B80CC8E2D043C077A1F12E457A75B1AD704091CA910316851E3855EB08D580BF8D
7DD15CAF656680630CD701786EC8BEFB5A33422F7CC4D78619D68542874D9916B962E3D7317AF118373480399D3508DA185394EA072C30240C2FA219F15A87ABF7E4046AAE90210C10AC9492E5A
EA7A26706C1356ADF8FDB6D5734C30E90CD632092E6B841EF26B87782EE2198DDBBBCEA0F217167D7991E67693D3D5D3EAAB1B51635847221A35A5FAF99ADA2A86E4B655AC507E58851835CA311
B29352FE09C66568F7CA260B0897CB0230F854BB5D00BA6AAE78A50EA0748811AE563754948371F0E574C0BFF1EF60B9859A612B0F3CEB462963B91D0C17B790F60662F868A82D7947C6D522799
80C9738C489259852B3D787FD001929D969B464F3AA570EE41708A7FB9AAE3599AAA19EF163038B5CDAE98CAA5BB79C3EC66EECB8D74332EE683B50BA64D2C7C04E2F23056F4D7CE02DF26417B6
3745BF38374193B077CFC3A817B2957E162EDB895C4311CA5B49B84BAA00BA4484BEBD43E86BD3A9B9D98B89A69C106CE2DCDA74ADD6C98AC39D09F077C6A4667A4AB4887CF6BB3C95D590A4C92
BD89A883A44D28E40BB89F9855C4FAD6FD15D6947279AB7B7DABCC3A5A35B7274EE6A4E12EE7979AAC2B2C01C4D9FA36C089A2CEA61DD1B71F611E3E0D9501BB820239453FD66C1D99F4E12EE69
CCF080E28AB311DEBDEAE869A4CBA7C31D7D846F7A103FEF6A358C571E3632B01482D75038C74BFD04DA323C60D7603F69420DDD7E6491E3ADBB8B0D94907CE982CA288C02F7962033903EC0016
AA81EBD7385F96FC890AE2C1A7EF4E4CE6E341A3E3E2567D1CBFE08378B4D0FF895D1757D266CC9BC0127DED3991ACD706A857CC853722A262AD3C9407E21BB0E3F97CFD588D73E73BCBA833151
77E9AC4BECA7B7D72ED2A5F7F7D3682A75C8AA51ACEA9F9A15E7C69B6B9E111379BFDE3B1B88EB24F0FF6563D05F686CAD9589DD231BC41489BF0169C7B46EE741EB8888BB870318F03FCA75930
4786B341EC267E82D59DF0D39D8638594F8D1DC355CB37862237FB7520949F119AE5C6FB2A0B535EFBBE91525DB19D06618B753A1C28439064FA2D20B640DB6C08E627B40F52AFD75F8F3727364
7509C5393571E1B4543DC5DEF5CAE7749B5A399CB2FC44AD1476FF827AFDE95D1241918405535ADCDE8AA9BCDA01CF204C1086F9D7B6AAF1E3512A7B29EE3C129DECE22F3B403CA665FA5804028
956551AD167958EFD1E4A89F4E7D85C7BD7600F9FB1958F66BA93FB68A2D2D7A506D65805D42FC893536D492584486F5FC4ADB63B9395DD0D0E36E07E7368D1FD0CF7B675E503683CDE26B7287A
E3C7C9135FD9EC8B5E510AA0DCBE362744AA9D54C4B1D8D46EBC01103DA3D82050702252EC9441D7CE8F224CF1219782F9ECF7E8D2EF96242B4353B3A91AD62125039554BA7F997AD0361175A0F
A4BAD6F0CBCC4EFD809AB372BA53AB5405131725F133984B2A25C60AF0212BD13D69ECCA50B43AE441BC7045D8252C886EA36BE444DBB48A77A3C48D1DF44E1995D7AD73966B7A92B629C9BB7DF
87D680C416D98DF5EDB06C0D1B7ED0324590A36BFD8FF5E2C715FFD0D49A3C0CA2C41D0215547956E852B59F62F57709011786A45FE0951FF7582B64F24F3E69637D2F37AFB87D1AA63F2BE2B47
831E9F11C3F186D4FE8EDF80136964A4A7BC73EDA83659E7D71185531A71567E00D6EF50BB91EEAAAB7998DFB6EA5CCF9B072B2817DDA5516891258CDB6D195C064BDB040478BD1211CD0F246B5
04B0708D992860B96110000CA110000504B01021E031400090008005C694549D992860B96110000CA1100000C00180000000000000000000A48100000000676C7970686167652E706E6755540500
03505EF55775780B000104F5010000041400000504B05060000000000010001005200000EC1100000000--496E2074686F736520646179732C20696E2074686F7365206661722072656D6F7465
20646179732C20696E2074686F7365206E69676874732C20696E2074686F73652066617261776179206E69676874732C20696E2074686F73652079656172732C20696E2074686F7365206661722
072656D6F74652079656172732C20617420746861742074696D6520746865207769736520666572077686F206B6E657720686F7720746F20737065616B20696E20656C61626F7261746520776F
726473206C6976656420696E20746865204C616E643B20437572757070617C20746865207769736520666E652C2077686F206B6E657720686F7720746F20737065616B2077697468206560C616
26F7261746520776F726473206C6976656420696E20746865204C616E642E20437572757070616720617661766520696E2073747275637469 6F6E7320746F2068697320736F6E3B20437572757061
672C2074686520736F6E206F662055626172612612D54757475206761766520696E737472756374696F6E7320746F2068697320736F6E205A692D75642D737572613A204D7920736F6E2C206C65742
06D65206769766520796F7520696E737472756374696F6E733A20796F752073686F756C642074616B6520617474656E74696F6E21205A692D75642D737572612C206C6574206D6520737065616B20
6120776F726420746F20796F753A20796F752073686F756C642074616B6520617474656E74696F6E2120446F206E6F74206E65676C656374206D7920696E737472756374696F6E732120446F206E6
F74207472616E7367726573732074686520776F7264732049207370656B21205468652069 6E737472756374696F6E73206F6620616E206F6C64206D616E206172652070726563696F75733B20
796F752073686F756C6420636F6D706C792077697468207468656D21AA546865206172746973746963206D6F75746820726563697465732074686520776F7264733B20746865206861727368206D6F75746
8206272696E6773206C697469676174696F6E20646F63756D656E74733B2074686520737765657420746F6E67756520676174686572732073776565742068657262732E--



+++ATH0