CS 188 Spring 2022 — Introduction to Artificial Intelligence — Midterm

- You have approximately 110 minutes.
- The exam is closed book, no calculator, and closed notes, other than a single two-sided "crib 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	
Exam Room	
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 a single two-sided crib 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: ______________________________

For staff use only:

Q1.	Potpourri	/14
Q2.	Ants: Escape!	/13
Q3.	Informed Search	/13
Q4.	Games	/18
Q5.	Just the right Bayes net	/10
Q6.	Sample Problem	/16
Q7.	Logical trades	/16
	Total	/100

THIS PAGE IS INTENTIONALLY LEFT BLANK

Q1. [14 pts] Potpourri

(a) [3 pts] Below is a list of task environments. For each of the sub-parts, choose all the environments in the list that falls into the specified type.
A: The competitive rock-paper-scissors game
B: The classical Pacman game (with ghosts following a fixed path)
C: Solving a crossword puzzle
D: A robot that removes defective cookies from a cookie conveyor belt

(i) [1 pt] Which of the environments can be formulated as *single-agent*? ☐ A ☐ B ☐ C ☐ D

(ii) [1 pt] Which of the environments are *static*? ☐ A ☐ B ☐ C ☐ D

(iii) [1 pt] Which of the environments are *discrete*? ☐ A ☐ B ☐ C ☐ D

(b) [2 pts]

(i) [1 pt] ○ T ○ F Reflex agents cannot be rational.

(ii) [1 pt] ○ T ○ F There exist task environments in which no pure reflex agent can behave rationally.

(c) [2 pts]

(i) [1 pt] ○ T ○ F If the costs can be arbitrarily large negative numbers in a search problem, then any optimal search algorithm in this problem will need to explore the entire state space.

(ii) [1 pt] ○ T ○ F Depth-first search always expands at least as many nodes as A* search with an admissible heuristic.

(d) [2 pts]

(i) [1 pt] ○ T ○ F Local beam search with a beam size of 1 reduces to Hill climbing.

(ii) [1 pt] ○ T ○ F Local beam search with one initial state and no limit on the number of states retained reduces to depth-first search.

(e) [4 pts]

(i) [1 pt] ○ T ○ F $A \Leftrightarrow B$ entails $\neg A \vee B$

(ii) [1 pt] ○ T ○ F α is satisfiable if and only if $\neg(\alpha \models \mathit{False})$

(iii) [1 pt] ○ T ○ F α is satisfiable if and only if $\neg(\mathit{False} \models \alpha)$

(iv) [1 pt] ○ T ○ F α is satisfiable if and only if $\neg(\mathit{True} \models \neg\alpha)$

(f) [1 pt] ○ T ○ F In a Bayes net, each node, given all of its parents and its children, is conditionally independent of all the other nodes in the graph.

Q2. [13 pts] Ants: Escape!

An ant wakes up and finds itself in a spider's maze!

- The maze is an M-by-N rectangle.
- Legal actions: $\{Forward, TurnLeft, TurnRight\}$.
- Transition model: *Forward* moves the ant 1 square in the direction it's facing, unless there is a wall in front. The two turning actions rotate the ant by 90 degrees to face a different direction.
- Action cost: Each action costs 1.
- Start state: The ant starts at (s_x, s_y) facing North.
- Goal test: Returns true when the ant reaches the exit at $G = (g_x, g_y)$.

(a) **(i)** [1 pt] What's the minimum state space size S for this task?

$S =$ ☐

(ii) [1 pt] Now suppose there are K ants, where each ant i must reach a distinct goal location G_i; any number of ants can occupy the same square; and **action costs are a sum of the individual ants' step costs**. What's the minimum state space size for this task, expressed in terms of K and S?

☐

(iii) [2 pts] Now suppose that each ant i can exit at any of the goal locations G_j, but no two ants can occupy the same square **if they are facing the same direction**. What's the minimum state space size for this task, expressed in terms of K and S?

☐

(iv) [2 pts] Now suppose, once again, that each ant i must reach its own exit at G_i, and no two ants can occupy the same square **if they are facing the same direction**. Let $H = \sum_i h_i^*$, where h_i^* is the optimal cost for ant i to reach goal G_i when it is the only ant in the maze. Is H admissible for the K-ant problem? Select all appropriate answers.

☐ Yes, because for any multiagent problem the sum of individual agent costs, with each agent solving a subproblem separately, is always a lower bound on the joint cost.
☐ Yes, because H is the exact cost for a relaxed version of the K-ant problem.
☐ Yes, because the "no two ants..." condition can only make the true cost larger than H, not smaller.
☐ No, because some ants can exit earlier than others so the sum may overestimate the total cost.
☐ No, it should be $\max_i$ rather than $\sum_i$.
○ None of the above

(b) The ant is alone again in the maze. Now, the spider will return in T timesteps, so the ant must reach an exit in T or fewer actions. Any sequence with more than T actions doesn't count as a solution.

In this part, we'll address this by solving the original problem and checking the resulting solution. That is, suppose p is a problem and A is a search algorithm; $A(p)$ returns a solution s, and $\ell(s)$ is the length (number of actions) of s, where $\ell(failure) = \infty$. Let p_T be p with the added time limit T. Then, given A, we can define a new algorithm $A'(p_T)$ as follows:

- $s \leftarrow A(p)$; **if** $\ell(s) \le T$ **then return** s **else return** *failure*.

(i) [1 pt] ○ T ○ F Suppose A is an optimal algorithm for p, action costs are 1; then A' is optimal for p_T.

(ii) [1 pt] ○ T ○ F Suppose A is a complete algorithm for p; then A' is complete for p_T.

(iii) [1 pt] ○ T ○ F Suppose A is an optimal algorithm for p, and action costs may be any nonnegative real number; then A' is optimal for p_T.

(c) Now we attempt to solve the time-limited problem by *modifying the problem definition* (specifically, the states, legal actions in each state, and/or goal test) appropriately so that regular, unmodified search algorithms will automatically avoid returning solutions with more than T actions.

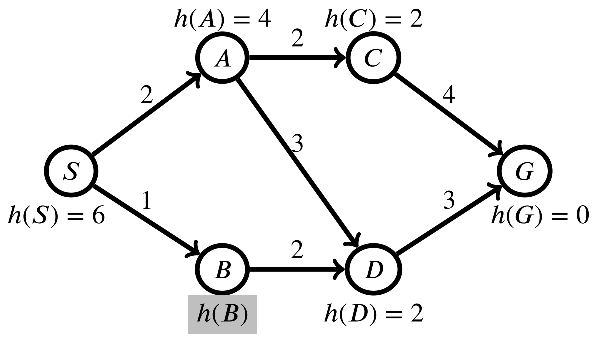
(i) [2 pts] Is this possible *in general, for any problem* where actions costs are all 1? Mark all correct answers.

☐ Yes, by augmenting the state space only.
☐ Yes, by augmenting the state space and modifying the goal test.
☐ Yes, by modifying the goal test only.
☐ Yes, by augmenting the state space and modifying the legal actions.
☐ Yes, by modifying the legal actions only.
◯ No, it's not possible in general.

(ii) [2 pts] Now suppose that instead of a bound T on the number of actions, there is a bound C on the total allowable cost, and that each action cost is at least ϵ, where $\epsilon > 0$. Is it possible to modify the problem definition so that regular, unmodified search algorithms will automatically avoid returning solutions with cost higher than C?

◯ Yes, and the state space can remain the same size.
◯ Yes, but the state space grows by a factor of C.
◯ Yes, but the state space may become infinite, even if the original state space is finite.
◯ No, it's not possible in general.

Q3. [13 pts] Informed Search



Search problem graph: S is the start state and G is the goal state.
Tie-break in alphabetical order.
$h(B)$ is unknown and will be determined in the subquestions.

(a) In this question, refer to the graph above where the optimal path is $S \to B \to D \to G$. For each of the following subparts, you will be asked to write ranges of $h(B)$. You should represent ranges as ___ $\leq h(B) \leq$ ___. *Heuristic values can be any number including* $\pm\infty$. *For responses of* $\pm\infty$, *you can treat the provided inequalities as a strict inequality.* If you believe that there is no possible range, please write "None" in the left-hand box and leave the right box empty.

(i) [2 pts] What is the range for the heuristic to be admissible?

☐ $\leq h(B) \leq$ ☐

(ii) [2 pts] What is the range for the heuristic to be consistent?

☐ $\leq h(B) \leq$ ☐

(iii) [2 pts] Regardless of whether the heuristic is consistent, what range of heuristic values for B would allow A* tree search to still return the optimal path ($S \to B \to D \to G$)?

☐ $\leq h(B) \leq$ ☐

(iv) [2 pts] Now assume that the edges in the graph are undirected (which is equivalent to having two directed edges that point at both directions with the same cost as before). Regardless of whether the heuristic is consistent, what range of heuristic values for B would allow A* tree search to still return the optimal path ($S \to B \to D \to G$)?

☐ $\leq h(B) \leq$ ☐

(b) Iterative deepening A* (IDA*) provides all the benefits of A* without needing to store all reached states in memory. In the following pseudocode, we provide an implementation of the IDA* algorithm. At each iteration, IDA* calls Distance-limited search which explores all nodes up to an f-cost limit. At the end of the iteration, the new limit value is updated to be the smallest f-cost that exceeds the current limit. Recall from A* search that the f-cost for a node is $f(n) = g(n) + h(n)$.

The pseudo-code for IDA* **tree search** is shown below.

```
function ITERATIVE-DEEPENING-A*-SEARCH(problem) return a solution node or failure
    limit ← f[start-node]
    while True do
        result, limit ← DISTANCE-LIMITED-SEARCH(problem, limit)
        if result ≠ "cutoff" then return result
function DISTANCE-LIMITED-SEARCH(problem, l) return a solution node or failure or cutoff and the new limit
    frontier ← a stack with NODE(problem, INITIAL) as an element
    result ← "failure"
    new-limit ← ∞
    while not IS-EMPTY(frontier) do
        node ← POP(frontier)
        if problem.IS-GOAL(node.STATE) then return node
        if f[node] > l then
            new-limit ← MIN(new-limit, f[node])
            result ← "cutoff"
        else
            if not IS-CYCLE(node) then
                for each child in EXPAND(problem, node) do
                    add child to frontier
    return result, new-limit
```

(i) [1 pt] Is IDA* tree search complete? ◯ Yes ◯ No

(ii) [1 pt] Is IDA* tree search optimal?

◯ IDA* is always optimal, regardless of the heuristic being used.
◯ IDA* is optimal if and only if the heuristic is admissible.
◯ IDA* is optimal if and only if the heuristic is consistent.
◯ IDA* is not optimal even if the heuristic is consistent.

(iii) [1 pt] b is the branching factor and d is the depth of the optimal solution. What is the space complexity of IDA* tree search if the initial cut-off threshold is smaller than d?

◯ $\mathcal{O}(b^d)$
◯ $\mathcal{O}(d^b)$
◯ $\mathcal{O}(bd)$
◯ None of above

(iv) [2 pts] If we maintain a closed set (also known as the *reached* set in AIMA Ch. 3) in the Distance-limited search function, we get Distance-limited **graph** search (the highlighted gray sections in the pseudo-code). However, the following command

closed-set ← INSERT(node, closed-set)

is missing. In this question, we will determine where the missing line of code should go.

```
1:  function ITERATIVE-DEEPENING-A*-SEARCH(problem) return a solution node or failure
2:      limit ← f[start-node]
3:      while True do
4:          result, limit ← DISTANCE-LIMITED-SEARCH(problem, limit)
5:          if result ≠ "cutoff" then return result
6:  function DISTANCE-LIMITED-SEARCH(problem, l) return a solution node or failure or cutoff and the new limit
7:      closed-set ← MAKE-EMPTY-SET()
8:      frontier ← a stack with NODE(problem, INITIAL) as an element
9:      result ← "failure"
10:     new-limit ← ∞
11:     while not IS-EMPTY(frontier) do
12:         node ← POP(frontier)
13:         __________(A)__________
14:         if node in closed-set then
15:             continue
16:         __________(B)__________
17:         if problem.IS-GOAL(node.STATE) then return node
18:         if f[node] > l then
19:             new-limit ← MIN(new-limit, f[node])
20:             result ← "cutoff"
21:         else
22:             if not IS-CYCLE(node) then
23:                 for each child in EXPAND(problem, node) do
24:                     add child to frontier
25:                 __________(C)__________
26:     return result, new-limit
```

Which of the following statements are true for IDA* using Distance-Limited Graph Search?

☐ If the command is added at line 13 (location marked (A)), the algorithm is optimal with a consistent heuristic.

☐ If the command is added at line 16 (location marked (B)), the algorithm is optimal with a consistent heuristic.

☐ If the command is added at line 16 (location marked (B)), the algorithm is complete but not optimal.

☐ If the command is added at line 25 (location marked (C)), the algorithm is complete but not optimal.

◯ None of the above

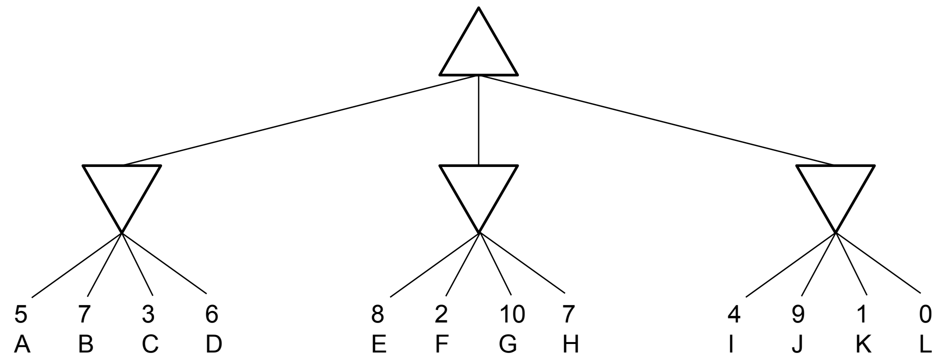
SID: ______________________________

Q4. [18 pts] Games

(a) Minimax and Alpha-Beta Pruning We have a two-player, zero-sum game with k rounds. In each round, the maximizer acts first and chooses from n possible actions, then the minimizer acts next and chooses from m possible actions. After the minimizer's k-th turn, the game finishes and we arrive at a utility value (leaf node). Both players behave optimally. Explore nodes from left to right.

(i) [1 pt] What is the total number of leaf nodes in the game tree, in terms of m, n, k? [ ]

(ii) [2 pts] In the minimax tree below $k = 1, n = 3, m = 4$.



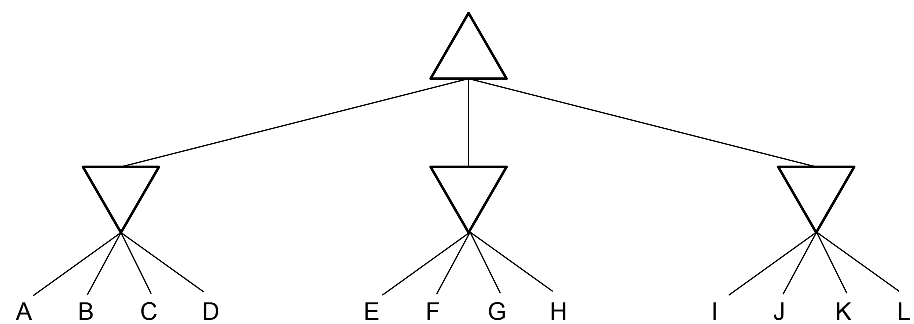
(1) What is the minimax value of the root? [ ]

(2) Which leaf nodes are pruned by alpha-beta? Mark the corresponding letters below.

☐ A ☐ B ☐ C ☐ D   ☐ E ☐ F ☐ G ☐ H   ☐ I ☐ J ☐ K ☐ L   ○ None

(iii) [1 pt] When $k = 1$, in the *best case* (pruning the most nodes possible), which nodes can be pruned in the tree below?

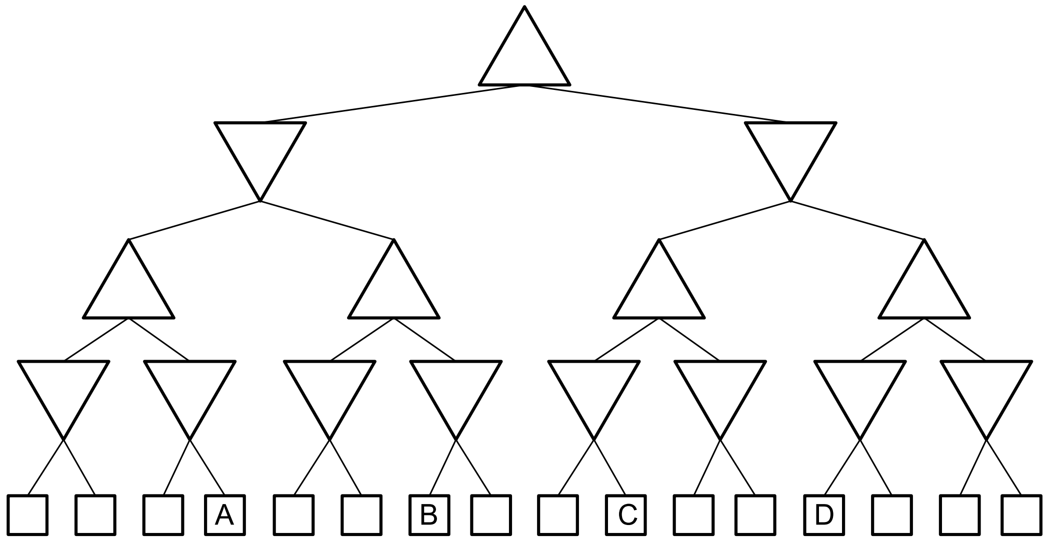
☐ A ☐ B ☐ C ☐ D   ☐ E ☐ F ☐ G ☐ H   ☐ I ☐ J ☐ K ☐ L   ○ None



(iv) [1 pt] Now consider the same $k = 1$ but with a general m and n number of maximizer and minimizer actions respectively. How many leaf nodes would be pruned in the best case? Express your answer in terms of m and n.

[ ]

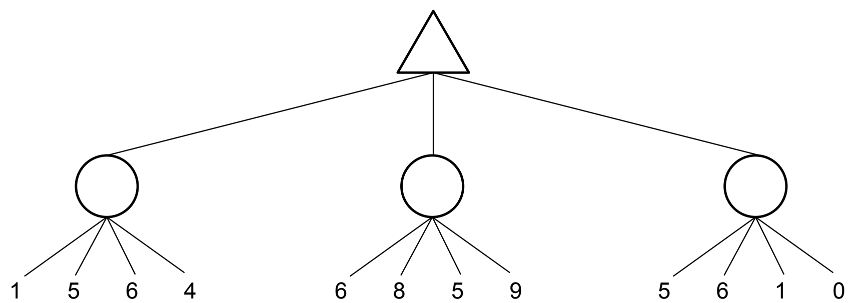
(v) [4 pts] When $k = 2, n = 2, m = 2$, in the best case, which of the leaves labeled A, B, C, D will be pruned?



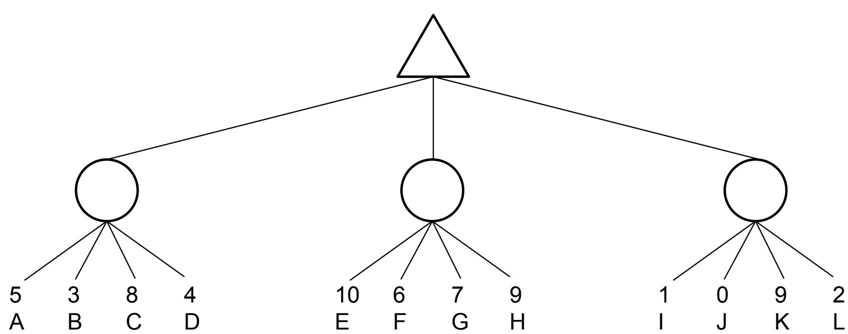
☐ A ☐ B ☐ C ☐ D ◯ None

(b) Chance Nodes Our maximizer agent is now playing against a non-optimal opponent. In each round, the maximizer acts first, then the opponent acts next and chooses uniformly at random from m possible actions.

(i) [1 pt] Consider the game tree below, with $m = 4$. What is the value of the root node? ☐



(ii) [2 pts] Consider the game tree below where we now **know** that the opponent always has $m = 4$ possible moves and chooses uniformly at random. We also know that all leaf node utility values are less than or equal to $c = 10$.



(1) What is the value of the root node? ☐

(2) Which nodes can be pruned?

☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G ☐ H ☐ I ☐ J ☐ K ☐ L ◯ None

(c) Now, let's generalize this idea for pruning on expectimax. We consider expectimax game trees where the opponent always chooses uniformly at random from m possible moves, and all leaf nodes have values no more than c. These facts are known by the maximizer player.

(i) [2 pts] Let's say that our depth-first traversal of this game tree is currently at a chance node and has seen k children of this node so far. The sum of the children seen so far is S. What is the largest possible value that this chance node can take on? (Answer in terms of m, c, k, and S)

(ii) [4 pts] Now, let's write an algorithm for computing the root value. Fill in the pseudocode below.

Note that m and c are constants that you should use in your pseudocode. To find the value at the root, we will start with a call to MAX-VALUE($root$, $-\infty$).

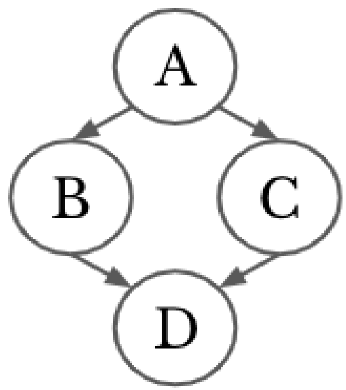
```
function MAX-VALUE(state, α)
    if state has no successors then return eval(state)
    v ← ________________________________
    for each successor n of state do
        v ← ________________________________
        ________________________________
    return v

function EXP-VALUE(state, α)
    if state has no successors then return eval(state)
    S ← 0
    k ← 1
    for each successor n of state do
        S ← S + ________________________________
        ci ← "expression from (c)(i) using m, c, k, and S"
        if ________________________________ then
            return S/m
        k ← k + 1
    return S/m
```

Q5. [10 pts] Just the right Bayes net

Bayes nets are used to represent probability distributions. We say that a given Bayes net structure B with variables $X_1, \ldots, X_n$ *can represent* a given target distribution $P(X_1, \ldots, X_n)$ if and only if there is some way to fill in the conditional distributions of B so that the global semantics are satisfied, i.e., $\prod_i P(X_i \mid Parents(X_i)) = P(X_1, \ldots, X_n)$. In particular, a Bayes net structure cannot represent a distribution P if it makes conditional independence assertions that do not hold in P.

(a) For this part, consider this Bayes net.



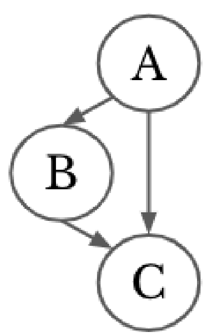
(i) [3 pts] Select all the conditional independences that are asserted by the network structure.

☐ A is conditionally independent of B given C.
☐ A is conditionally independent of B given D.
☐ B is conditionally independent of C given A.
☐ D is conditionally independent of A given B.
☐ D is conditionally independent of A given B and C.
○ None of the above

(b) You go to discussion, and afterwards you discuss Bayes nets with friends (so studious!).

Let's evaluate each of their claims.

(i) [1 pt] Jocelyn claims that her favorite Bayes net (below) can encode all possible distributions of three variables. Would you agree?



○ Yes, because the Bayes net is acyclic
○ Yes, for a reason other than above
○ No, because reversing the arrow between B and C would give a different structure
○ No, for a reason other than above

(ii) [1 pt] Jason mentions that he thinks that adding an edge to a Bayes net will always strictly increase the number of distributions the Bayes net can represent. Is he right?

◯ He's right, because adding an edge means there are fewer conditional independences in the Bayes net structure

◯ He's right, for a reason other than above

◯ He's wrong, since adding an edge will decrease the number of distributions a Bayes net can encode

◯ He's wrong, for a reason other than above

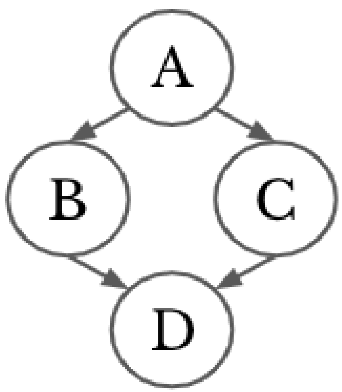
(iii) [2 pts] Angela claims that every Bayes net with the same number of edges has the same number of independences. Is she right?

◯ She's right, because removing any arc removes the same number of conditional independences

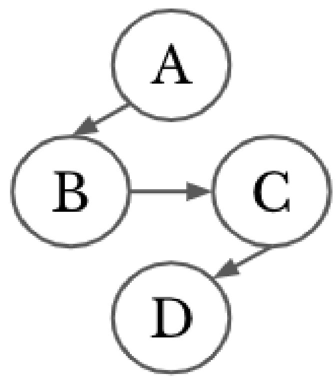
◯ She's right, for a reason other than above

◯ She's wrong

(c) [3 pts] Given that a distribution $P(A, B, C, D)$ can be represented by *both* of the following Bayes nets, what do we know for sure about P?



Bayes net 1



Bayes net 2

☐ In the distribution, A is conditionally independent of D given C

☐ A and B cannot be independent

☐ The distribution has all the conditional independences that hold in both Bayes nets

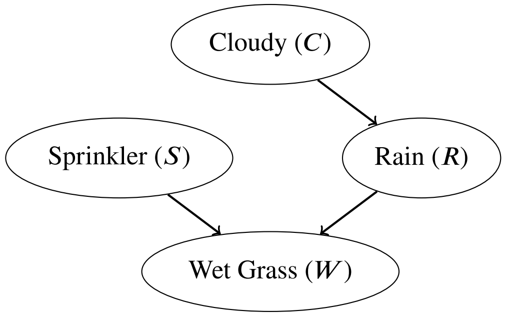
☐ The distribution has all the conditional independences that hold in either Bayes net 1 or Bayes net 2

☐ There is no distribution that can be represented by both Bayes nets

◯ None of the above

Q6. [16 pts] Sample Problem

Recall the following Bayes net adapted from lecture where we have **binary** variables Cloudy (C), Rainy (R), Sprinkler (S), and Wet Grass (W); except unlike in lecture, the sprinkler acts independently of the cloudiness (as depicted below). Note that the domain for each variable X is $\{x, \neg x\}$.



Sample number	C	S	R	W
1	c	s	r	w
2	c	s	r	w
3	$\neg c$	$\neg s$	r	w
4	$\neg c$	$\neg s$	r	$\neg w$
5	$\neg c$	$\neg s$	r	$\neg w$

(a) **(i)** [3 pts] Consider the sequence of samples shown above on the right. For each of the following sampling methods, select all the samples that could be generated to determine $P(C = c | R = r, W = w)$.

(1) Prior Sampling: ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ○ None

(2) Rejection Sampling: ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ○ None

(3) Likelihood Weighting: ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ○ None

(ii) [1 pt] According to the order that samples appear in the table, select the sample numbers forming the longest consecutive sequence of samples that could appear using Gibbs sampling to approximate $P(C = c | R = r, W = w)$. (For example, if you think Gibbs sampling could generate samples 2, 3, 4, 5 but not 1, mark 2, 3, 4, 5.)

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ○ None

(iii) [1 pt] *True/False*: With an identical and finite set of samples (not necessarily the set of samples shown above) that is valid for all the sampling methods mentioned in (i) and (ii), all of the sampling methods will return the same result for $P(C = c | R = r, W = w)$.

○ T ○ F

(iv) [2 pts] You are performing Gibbs sampling on the above Bayes net to calculate $P(C = c | R = r, W = w)$ and decide to sample the variable S in the next iteration. Which of the following CPTs are required to calculate the probability of sampling $S = s$?

☐ $P(C)$
☐ $P(R|C)$
☐ $P(S)$
☐ $P(W|S, R)$

(b) [1 pt] Alice wants to determine $P(C = c|evidence)$ but she cannot sample S because $P(S)$ is missing. Which of the following queries could still be answered?

☐ $P(C = c | R = r, W = w)$
☐ $P(C = c | W = w)$
☐ $P(C = c | R = r)$
○ It is impossible to perform approximate inference without sampling S.

(c) Bob wants to perform likelihood weighting on $P(C = c|W = w)$ but he knows that it may be inefficient with downstream evidence such as $W = w$. With his knowledge of equivalent Bayes net representations, he proposes the method of Reversed Likelihood Weighting:

1. Construct an equivalent reversed Bayes net.
2. Calculate the new CPTs using Bayes' rule.
3. Perform likelihood weighting on the modified Bayes net.

To test his theory, consider a Bayes net with two variables, A and B, shown on the left where $B = b$ is given as evidence and the query is $P(a|b)$. The reversed Bayes net is shown on the right.

(a) Original Bayes net (b) Reversed Bayes net

(i) [3 pts] Using only the CPT entries of the original problem ($P(A)$ and $P(B|A)$), write the expression Bob would use to calculate each new CPT of the reversed Bayes net.

(1) $P(B) =$ [ ]

(2) $P(A|B) =$ [ ]

(ii) [1 pt] Using only the CPT entries of the reversed Bayes net, write an expression for the weight that would be assigned to a sample (a, b) in Reversed Likelihood Weighting.

[ ]

(iii) [1 pt] Is Reversed Likelihood Weighting a consistent sampling method?

- ◯ Yes because the new Bayes net represents the same joint distribution, and likelihood weighting on the new Bayes net is consistent.
- ◯ Yes but not for the reason above.
- ◯ No because reversing the edge changes the conditional probability terms used in likelihood weighting to calculate the probability of a sample.
- ◯ No but not for the reason above.

(iv) [1 pt] *True/False:* Define the *weighted probability* of a complete sample as the probability of generating that sample times the weight it would be assigned. Then for any given sample generated from the two networks shown above, the weighted probability of the sample in each case is identical.

◯ T ◯ F

(v) [2 pts] Assume that, just by chance, we have generated the *same set of 100 samples* from the original graph and the reversed one. Will original likelihood weighting and Reversed Likelihood Weighting yield the same result for $P(a|b)$?

- ◯ Yes because the weights for each sample are the same for the original and reverse cases.
- ◯ Yes because the weighted probability of each sample is the same for the original and reverse cases.
- ◯ Yes but not for the reasons above.
- ◯ No because the weights for each sample are different between the original and reverse cases.
- ◯ No because the weighted probability of each sample is different between the original and reverse cases.
- ◯ No but not for the reasons above.

Q7. [16 pts] Logical trades

This question takes the first steps in using first-order logic to formalize a world in which people own and trade cards based on their preferences. The basic vocabulary is as follows:

- $Likes(p, c)$: person p likes card c.
- $Owns(p, c)$: person p owns card c.
- $Prefers(p, c_1, c_2)$: person p (strictly) prefers card c_1 to card c_2.
- $Indifferent(p, c_1, c_2)$: person p is indifferent between card c_1 and card c_2.
- $Tradeable(p_1, c_1, p_2, c_2)$: true if and only if p_1 would trade c_1 to p_2 in return for c_2.

Important notation: To keep the logical sentences concise, we will use the convention that logical variables beginning with p are assumed to refer to people and variables beginning with c are assumed to refer to cards. Thus, for example, $\forall p\ Happy(p)$ is equivalent to the more complete version, $\forall p\ Person(p) \Rightarrow Happy(p)$.

(a) In this part you will check some axioms that claim to capture the properties of the terms above. For each, mark True/False if the logical sentence is a correct expression of the fact described in English.

(i) [1 pt] Everyone likes at least one card.

○ T ○ F $\forall p\ \exists c\quad Likes(p, c)$

(ii) [1 pt] No one prefers a card they don't like to a card they like.

○ T ○ F $\forall p, c_1, c_2\quad Likes(p, c_2) \Rightarrow [Likes(p, c_1) \vee \neg Prefers(p, c_1, c_2)]$

(iii) [1 pt] Everyone prefers cards they like to cards they don't like.

○ T ○ F $\forall p, c_1, c_2\quad Likes(p, c_1) \wedge \neg Likes(p, c_2) \Rightarrow Prefers(p, c_1, c_2)$

(iv) [1 pt] No two people own the same card.

○ T ○ F $\forall p_1, p_2, c\quad \neg Owns(p_1, c) \vee \neg Owns(p_2, c)$

(v) [1 pt] A person cannot prefer one card to another and vice versa.

○ T ○ F $\forall p, c_1, c_2\quad \neg Prefers(p, c_1, c_2) \vee \neg Prefers(p, c_2, c_1)$

(vi) [2 pts] A person either prefers one card to another, or is indifferent between them, but not both.

○ T ○ F $\forall p, c_1, c_2 [[Prefers(p, c_1, c_2) \vee Prefers(p, c_2, c_1)] \Leftrightarrow \neg Indifferent(p, c_1, c_2)]$
$\wedge [Prefers(p, c_1, c_2) \vee Prefers(p, c_2, c_1) \vee Indifferent(p, c_1, c_2)]$

(b) [2 pts] ○ T ○ F Assuming both were written correctly, the sentence in a(ii) entails the sentence in a(iii).

(c) [3 pts] Regardless of whether they are correct expressions for the English sentences, which of the logical sentences in (a), when converted to CNF, yield only *definite clauses*, i.e., clauses with exactly one positive literal?

☐ (i) ☐ (ii) ☐ (iii) ☐ (iv) ☐ (v) ☐ (vi) ○ None

(d) [4 pts] Write a sentence in first-order logic to express a fact about *Tradeable*: people will trade cards if and only if each prefers the card the other person owns to the one they own.

SID: ____________________________

Q1

(a) **(i)** ☐ **(ii)** ☐ **(iii)** ☐

(b) **(i)** ☐ **(ii)** ☐

(c) **(i)** ☐ **(ii)** ☐

(d) **(i)** ☐ **(ii)** ☐

(e) **(i)** ☐ **(ii)** ☐ **(iii)** ☐ **(iv)** ☐

(f) **(i)** ☐

Q2

(a) **(i)** ☐ **(ii)** ☐ **(iii)** ☐

(iv) ☐

(b) **(i)** ☐ **(ii)** ☐ **(iii)** ☐

(c) **(i)** ☐ **(ii)** ☐

Q3

(a) **(i)** **(1)** ☐ **(2)** ☐

(ii) **(1)** ☐ **(2)** ☐

(iii) **(1)** ☐ **(2)** ☐

(iv) **(1)** ☐ **(2)** ☐

(b) **(i)** ☐ **(ii)** ☐ **(iii)** ☐

(iv) ☐

Q4

(a) (i)

(ii) (1) (2)

(iii)

(iv) (v)

(b) (i)

(ii) (1) (2)

(c) (i)

(ii) (1)

(2)

(3)

(4)

(5)

Q5

(a) (i)

(b) (i) (ii) (iii)

(c)

Q6

(a) (i) (1) (2) (3)

(ii)

(iii) (iv)

(b) [ ]

(c) **(i)** **(1)** [ ] **(2)** [ ]

(ii) [ ]

(iii) [ ] **(iv)** [ ] **(v)** [ ]

Q7

(a) **(i)** [ ] **(ii)** [ ] **(iii)** [ ] **(iv)** [ ] **(v)** [ ] **(vi)** [ ]

(b) **(i)** [ ]

(c) **(i)** [ ]

(d) **(i)** [ ]