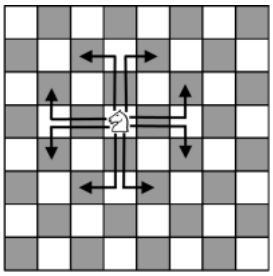
National University of Singapore
School of Computing
CS3243 Introduction to AI

Final Quiz (Solutions)

1. (**AY 2019/2020 ST 1 Final Exam Question - Uninformed Search**)
In International Chess, the Knight piece is able to move as shown in the figure below.



Given a chessboard of arbitrary size, $n \times n$ (e.g., the chessboard in Figure 1 is an 8×8 board), let a Knight's Tour be defined as follows:

A knight is positioned at some starting position (ROW, COL), where $1 \leq$ ROW, COL $\leq n$ To complete the tour, it must visit every square on the board (i.e., every position (r, c) where $1 \leq r, c \leq n$) **exactly once**.

(a) Model the Knight's Tour as an Uninformed Search Problem. Clearly define the states, actions, transition model, goal test and cost function.

(b) Assuming that you could only choose between Breadth-first Search and Depth-first Search, pick one and explain why it is the better choice.

(c) Would you consider using the Iterative-Deepening Search algorithm? Provide a rationale for your answer.

Solution:

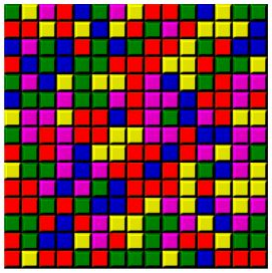
(a) **State** contains position of the knight, visited positions.
Initial state: knight at starting position (Row, Col), and visitedSet containing the start position as well.
Actions: 8 possible moves that a knight can make.
Transition: displacement set (-2, 1), (-2, -1), (2, 1), (2, -1), (1, 2), (-1, 2), (-1, 2), (-1, -2). Resultant position of the knight is x, y = curr_i + dx, curr_j + dy for dx, dy in displacement set. A valid move is when the resultant position of the knight is on

the board ($1 \leq x, y \leq n$) and the position have not been visited before ((x, y) not in visitedSet). New state will be knight position = x, y where x, y is the resultant position of the move, and (x, y) is added into the visitedSet
Goal test: size of visitedSet = 64
Cost function: Each move cost 1.

(b) Depth First Search. The solution is constant at some depth.

(c) No, DFS will suffice. All solutions lie at specific depth IDS will do unnecessary extra work.

2. (**AY 2019/2020 ST 1 Final Exam Question - Informed Search**)
In the SameGame puzzle, a player is given a two-dimensional, rectangular, $n \times m$ grid of coloured squares. An example of such a grid is depicted in the figure below.



The grid is initially filled with $n \times m$ blocks with c different colours. The position of a grid/tile location is denoted (ROW, COL), where $1 \leq$ ROW $\leq n$ and $1 \leq$ COL $\leq m$. Further, let the position (1, 1) reference the top left corner of the grid.

Two tiles are directly adjacent if they are horizontal or vertical direct "neighbors". A tile (i, j) not on the edge of the board has neighbors $(i-1, j), (i+1, j), (i, j-1)$ and $(i, j+1)$, while a tile on the edge has only three of these neighbors, and a corner tile only two.

A group is defined as a set of two or more tiles of the same color, where it is possible to go from any of its tiles to any other by repeatedly visiting adjacent tiles; or briefly: a group is connected. A tile at the board that does not belong to any group is called a singleton.

A move consists of deleting a group (not a singleton), after which, vertical gravity and column shifting are applied: i.e., tiles that are above the deleted tiles fall down, and when any column is completely empty, the columns to the right of the empty column(s) shift to the left.

The goal is to empty the board, leaving no tiles. We call a puzzle solvable if the board can be emptied by doing a series of moves.

Let the states in this problem correspond to different tile compositions within the $n \times m$ grid, with the initial state corresponding to some initial placement of nm tiles, with each tile corresponding to one of the possible c colours (an example is given in the figure above).

An action is a legal removal of a group of tiles, with the transition model applying vertical gravity and column shifting. The goal state is an empty grid. The transition cost is 1, except for grids where no groups can be removed, in which case the cost of all actions is $\inf$.

Design an admissible heuristic for this puzzle game. Your heuristic may not be $h(s) = 0$ for all states s, the (abstract) optimal heuristic, or a linear combination/simple function thereof. You may assume that the tile layout in the initial grid is solvable i.e. there is some path to a goal state. **You must prove that your heuristic is admissible.**

Solution: $h(s)$ = number of colours remaining. Proof of admissibility: Each group contains exactly 1 colour. For each remaining colour, 1 or more groups. Each move may reduce groups by more than 1 but never reduce remaining groups of 1 colour below 1 (unless that colour is removed by the move).

3. (**AY 2019/2020 ST 1 Final Exam Question - Adversarial Search**)
Suppose that you are given an MDP with a state-dependent reward Which of the following statement is true when all the terminal nodes of a game tree is multiplied by a constant k?

   (a) The minimax value at the root will be multiplied by k.

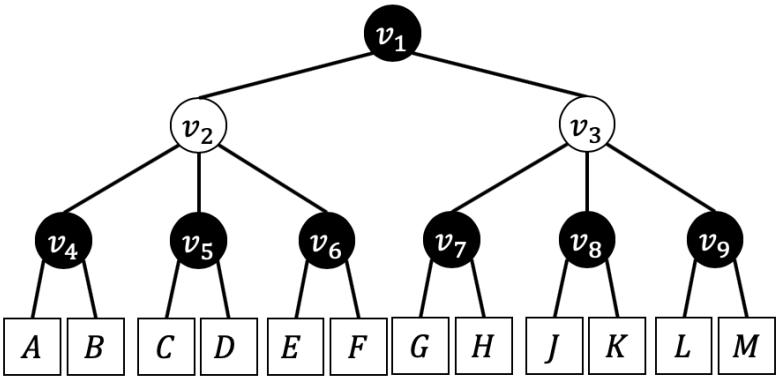
   (b) The minimax value at the root will be multiplied by $\frac{1}{k}$, only if $k \neq 0$.

   (c) The minimax value at the root will remain unchanged.

   (d) The minimax value at the root will change, but the change cannot be predicted with certainty.

   Solution: The answer is (d). When $k \geq 0$, the minimax value at the root will be multiplied by k. When $k \leq -1$, it is not the case.

4. (**AY 2019/2020 ST 1 Final Exam Question - Adversarial Search**)
Black nodes correspond to MAX player, white nodes corresponds to MIN player.



Given we run the alpha-beta pruning algorithm on the above tree from left to right, which of the following is true?

You may assume $A, B, C, D, E, F, G, H, J, K, L, M$ are positive integers.

(a) There exist some set of values at the leaf nodes such that node B is not evaluated.

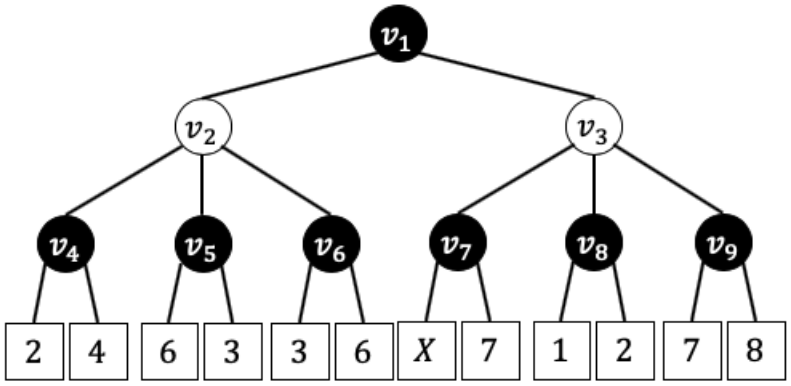
(b) There exist some set of values at the leaf nodes such that node F is not evaluated.

(c) There exist some set of values at the leaf nodes such that both nodes G and H are not evaluated.

(d) All nodes will be evaluated no matter what values are set at the leaf nodes.

Solution: The answer is (b).

5. (**AY 2019/2020 ST 1 Final Exam Question - Adversarial Search**)
Black nodes correspond to MAX player, white nodes corresponds to MIN player.



Given we run the alpha-beta pruning algorithm on the above tree from left to right, answer the questions that follow.

(a) What is the maximum integer value the leaf node labelled X can take on (assume the range $1 \leq X \leq 100$), such that minimal arcs are pruned?

(b) Let the leaf node labelled X take on the value you gave in (a), what is the minimax value at the root?

(c) Let the leaf node labelled X take on the value you gave in (a), how many leaf nodes are not evaluated?

Solution:

(a) 100

(b) 4

(c) 3

6. (**AY 2019/2020 ST 1 Final Exam Question - Constraint Satisfaction Problems**)
Hanabi is a cooperative card game in which a set of players are aware of other players' cards but not their own, attempt to play a series of cards in a specific order to set off a simulated fireworks show.

In this question, we will model some elements of agents' initial hands and the playing field as constraints of a constraint satisfaction problem.

The Hanabi deck contains cards $c \in C$ from five possible suits S = {white, yellow, green, blue, and red} and numbers $N = \{1, 2, 3, 4, 5\}$, with quotas: three 1's, two each of 2's, 3's, and 4's, and one 5. The suit (also known as colour), s, of a card c can be evaluated using the function suit(c, s), which returns 1 if the card is of that suit, and 0 otherwise; and the number n of a card c can be evaluated using the function number(c, n), which returns 1 if the card is of that number, and 0 otherwise.

The playing field (to "build fireworks") consist of five (initially empty) piles of cards, each corresponding to a firework of a unique suit (each suit can only have exactly one pile). Denote the jth ($j = 1, 2, 3, 4$ or 5) card of a pile of suit s as `firework`(s, j). If there's no card at index j, then the function returns `NULL`; else, it returns a card. The goal of the players is to build as many firework piles as possible, while maintaining a valid firework display. We assume that when a player plays a card to a deck, it automatically fills the lowest empty index slot on the pile.

- `ValidFireworkDisplay` constraint: a valid firework display is where for every firework pile of suit s, the jth index of the firework pile of suit s contains the card c of suit s and number j, if `firework`(s, j) is non-empty.

To start the game, each player $i \in Q$ is dealt a hand $H_i = \{c_{i1}, c_{i2}, c_{i3}, c_{i4}, c_{i5}\}$ containing five cards, where $\{c_{i1}, c_{i2}, c_{i3}, c_{i4}, c_{i5}\} \in C$.

- `ValidHandSuit` constraint: all players' hands are valid (suit-wise) if for every player i, it respects the initial quantity of cards' suits in the deck (e.g. there should not be eleven red cards in a player's hand at any point in time, because only ten red cards exists in the deck!)
- `ValidHandNumber` constraint: all players' hands are valid (number-wise) if for every player i, it respects the initial quantity of cards' number in the deck (e.g. there should not be five 5's in a player's hand at any point in time, because only four 5 exists in the deck!)

In the questions that follow, you may only use standard mathematical operators $(+, -, \times, \div)$ and standard logical and set operators $(\forall, \exists, \vee, \wedge$ and $x \in X, X \subseteq Y)$. You may use English, but it has to be unambiguously directly convertible into logical symbols.

(a) Write the `ValidFireworkDisplay` constraint. You may assume you are given the function `empty`(s, j), which returns `true` if there is no card at index j on the fireworks pile belonging to suit s. You may also use any given functions in the question.

(b) Write the `ValidHandSuit` constraint. You may assume you are given the function `quantity`(s), which returns the quantity of cards in the initial deck of cards of a particular suit s. You may also use any given functions in the question.

(c) Write the `ValidHandNumber` constraint. You may assume you are given the function `quantity`(n), which returns the quantity of cards in the initial deck of cards of a particular number n. You may also use any given functions in the question.

Solution:

(a) for all colour s in S, and j from 1 to 5, not `empty`(s, j) implies [`number(firework`(s, j)) $= j$ and `colour(firework`(s, j)) = s]

(b) for all i in Q, for all s in S, sum over c in H_i(`suit`(c, s)) <= `quantity`(s)

(c) for all i in Q, for all n in N, sum over c in H_i(`number`(c, n)) <= `quantity`(n)

7. (**AY 2019/2020 ST 1 Final Exam Question - Constraint Satisfaction Problems**)
Which of the following is NOT true about the AC-3 algorithm? You may assume n is the number of variables, and each variable's domain has a maximum of d values.

(a) The AC-3 algorithm has an overall time complexity of $O(n^2 d^3)$.

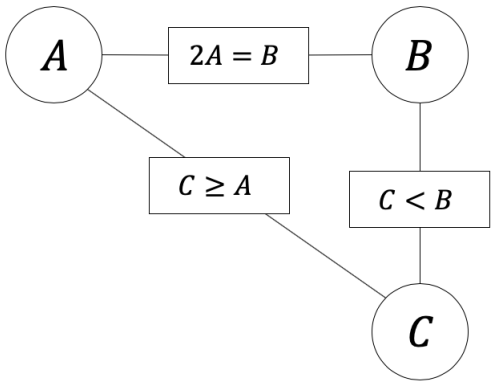
(b) Checking the consistency of an arc takes $O(d)$ time.

(c) The AC-3 algorithm can be used as a pre-processing step before running the backtracking algorithm for solving a CSP.

(d) Whenever the domain of a variable A is reduced to maintain arc-consistency of (A, B), we do not add (B, A) to the queue.

Solution: (b) is incorrect. Checking the consistency of an arc in AC-3 takes $O(d^2)$ time.

8. **AY 2019/2020 ST 1 Final Exam Question - Constraint Satisfaction Problems**



(a) Trace the AC-3 algorithm on the above constraint graph.
Initially, the domain of each variable is $D_A = D_B = D_C = \{1, 2, 3, 4\}$
Assume that the initial queue is: $(A, B), (B, A), (B, C), (C, B), (C, A), (A, C)$; where (A, B) is at the head of the queue.
Your answer should be of the form (just an example):
1. Domain of X reduced to $\{5, 6, 7, 8\}$, queue: $(Y, X), (Z, W)$
2. Domain of Y not reduced, queue: (Z, W)
...
where the variable, its domain, and the queue after this processing is clearly shown.
Note: if the arc is already in the queue, do NOT add it to the queue again.

(b) With reference to the previous question, provide a valid assignment of values to A, B, and C such that the constraints are satisfied, and $A + B + C$ is minimum.

Solution:

(a) Initial queue: $(A, B), (B, A), (B, C), (C, B), (C, A), (A, C)$
1. Domain of A reduced to $\{1,2\}$, queue: $(B, A), (B, C), (C, B), (C, A), (A, C)$
2. Domain of B reduced to $\{2,4\}$, queue: $(B, C), (C, B), (C, A), (A, C)$

3. Domain of B not reduced, queue: $(C, B), (C, A), (A, C)$
4. Domain of C reduced to {1,2,3}, queue: $(C, A), (A, C)$
5. Domain of C not reduced, queue: (A, C)
6. Domain of A not reduced

(b) $A = 1, B = 2, C = 1$

9. (**AY 2019/2020 ST 1 Final Exam Question - Reinforcement Learning**)
Suppose you are given two algorithms for a reinforcement learning problem. Algorithm 1 returns the optimal state value function ($V^*(s)$) for the problem, while Algorithm 2 returns the optimal action value function ($Q^*(s, a)$). Assuming that you do not know the state transition probabilities for this problem, which algorithm would you prefer for designing an agent that can act optimally?

In the rationale box, briefly and clearly explain why you would choose one algorithm over the other.

Solution: Algorithm 2 is the preferred one. It gives you $Q^*(s, a)$, and hence, you can design the agent's policy by selecting $\pi^*(s) = \text{argmax}_a(Q(s, a))$. However, this is not possible with $V^*(s)$, since you do not know the state transition probabilities. Thus, given $V^*(s)$, you cannot be sure which action is optimal at state s.

10. (**AY 2019/2020 ST 1 Final Exam Question - Reinforcement Learning**)
Consider a 9×9 grid world, where an agent starts each episode in the center square. The goal of the agent is to reach the top-right square in the minimum number of steps (given the usual left/right/up/down moves). You decide to use the following reward formulation in order to find the optimal policy for this problem: the agent receives a reward of +5 on reaching the goal state, and no reward for any other transition.

Suppose, you try three variants of the above reward formulation with the Q-learning algorithm: A_1 with $\gamma \in (0, 1)$, A_2 with $\gamma = 0$ and A_3 with $\gamma = 1$ (where γ denotes the discount factor). Select all of the following that can be concluded with certainty. If none of the options are correct, leave all options unselected.

(a) A_1 learns the desired policy.

(b) A_2 learns the desired policy.

(c) A_3 learns the desired policy.

In the rationale box, briefly and clearly explain why each option is correct/incorrect. Note that you need to justify for all cases.

Solution:

(a) Correct. With $\gamma \in (0, 1)$, the agent learns the optimal policy since future rewards are discounted, and getting to goal node soonest gives the highest total rewards (discounted).

(b) Incorrect. With $\gamma = 0$, all future rewards carry zero value. Hence, we cannot conclude with certainty that the optimal policy will be learned.

(c) Incorrect. With $\gamma = 1$, it does not matter how many steps the agent takes to reach the goal node, since it always gets a reward of +5 on reaching the goal. Hence, we cannot conclude with certainty that the optimal policy will be learned.

11. (**AY 2019/2020 ST 1 Final Exam Question - Reinforcement Learning**)
In order to train an RL agent to play the game of golf using a simulator, you are given a reward function such that the agent receives a reward of +10 when the golf ball is hit into the hole, and -1 for all other transitions. An episode terminates when the ball is hit into the hole (goal state). Your want to reach the goal state in the minimum number of hits. In order to help the agent in learning, you decide to provide an additional reward of +5 whenever the ball is within a 0.5 metre radius of the hole. You also choose $\gamma = 1$. Assuming that you have access to an algorithm that can generate the optimal policy given the problem and the reward formulation, which of the following options is/are correct?

(a) The additional reward helps the agent by nudging it towards a sub-goal, and thus, the agent will learn quicker to play golf.

(b) The additional reward will slow down the learning process, although the agent will still be able to learn the desired policy for playing golf.

(c) The additional reward may cause the agent to learn undesired behaviour.

(d) Your conclusion changes when $\gamma < \frac{1}{2}$

In the rationale box, briefly and clearly justify the option(s) you select. Note that you do not need to provide a rationale for the option(s) you do not select.

Solution:

(a) Incorrect.

(b) Incorrect.

(c) Correct. The additional reward may cause the agent to learn undesired behaviour. This is because the agent will now try to keep the ball within a 0.5m radius to keep collecting rewards, rather than terminate by putting the ball into the hole.

(d) Correct. Your conclusion changes when $\gamma < \frac{1}{2}$. This is because with discounting, the optimal policy becomes the one which puts the ball into the hole in a single move, rather than moving the ball around the hole within a 0.5m radius for a certain number of steps (possibly infinite), before putting the ball into the hole (possibly never).

12. (**AY 2019/2020 ST 1 Final Exam Question - Reinforcement Learning**)
If $Q^\pi(s, a) > V^\pi(s)$, which of the following can be concluded with certainty?

(a) a is the best action at state s

(b) π may be an optimal policy

(c) π is not an optimal policy

(d) a is not the best action at state s

In the rationale box, briefly and clearly justify the option(s) you select. Note that you do not need to provide a rationale for the option(s) you do not select.

Solution:

(a) Incorrect.

(b) Incorrect.

(c) Correct. π is not an optimal policy (since $Q^\pi(s, a) > V^\pi(s)$ implies taking action a at state s gives better rewards than choosing the action given by π at state s).

(d) Incorrect.

13. (**AY 2019/2020 ST 1 Final Exam Question - Reinforcement Learning**)
True/False: Suppose, you add a constant $C > 1$ to all rewards in an **infinite horizon** MDP (where there are no terminating states). An optimal policy under the earlier MDP will remain an optimal policy under the transformed MDP.

In the rationale box, briefly and clearly explain your choice.

Solution: True. Since it is an infinite horizon MDP, adding constants to all rewards has no effect on the optimal policy, since the relative order of value functions of the states stays the same.

14. (**AY 2019/2020 ST 1 Final Exam Question - Reinforcement Learning**)
True/False: Suppose, you add a constant $C > 1$ to all rewards in a **finite horizon** MDP (where there are terminating states). An optimal policy under the ealier MDP will remain an optimal policy under the transformed MDP.

In the rationale box, briefly and clearly explain your choice.

Solution: False. Since it is a finite horizon MDP, a policy which was optimal under the earlier MDP might not be optimal any more. Easy to construct examples to show this, such as the gridworld in Q10 with $\gamma \in (0, 1)$.

15. (**AY 2019/2020 ST 1 Final Exam Question - Logic and Logical Agents**)
True/False: The knowledge base of an agent cannot change.

In the rationale box: justify your answer.

Solution: False. The knowledge base of an agent can change e.g. when it infers new things via its inference engine or it learns new information via its percepts.

16. (**AY 2019/2020 ST 1 Final Exam Question - Logic and Logical Agents**)
Consider the following knowledge base:
"All firetrucks are red"
"All firetrucks are cars"
"All cars have four wheels"

An inference algorithm that gets the sentence "A ferrari is a red car" and infers "A ferrari is a firetruck" cannot be:

(a) complete

(b) sound

(c) both of the above

In the rationale box: justify your answer.

Solution: (b) is the correct answer. As the inferred statement is not entailed by the knowledge base the algorithm cannot be sound. It could still be complete, as the question doesn't state if the algorithm inferred anything else.

17. **AY 2019/2020 ST 1 Final Exam Question - Logic and Logical Agents**
Consider the following knowledge base:
"All firetrucks are red"
"All firetrucks are cars"
"All cars have four wheels"

Given the knowledge base, which of the properties would guarantee that an algorithm can infer that ferraris have four wheels from the fact that they are red cars:

(a) completeness

(b) soundness

(c) both of the above need to be combined

In the rationale box: justify your answer.

Solution: (a) is the correct answer. Completeness ensures that everything that is entailed by the knowledge base will be inferred, as the statement is entailed by the KB completeness guarantees its inference.

18. (**AY 2019/2020 ST 1 Final Exam Question - Logic and Logical Agents**)
Choose which of the following is an example (factually correct) of entailment:

In the rationale box: justify your answer.

(a) All birds are animals

(b) Most birds can fly

(c) All flying animals are birds

(d) Insects and birds have wings

In the rationale box: justify your answer.

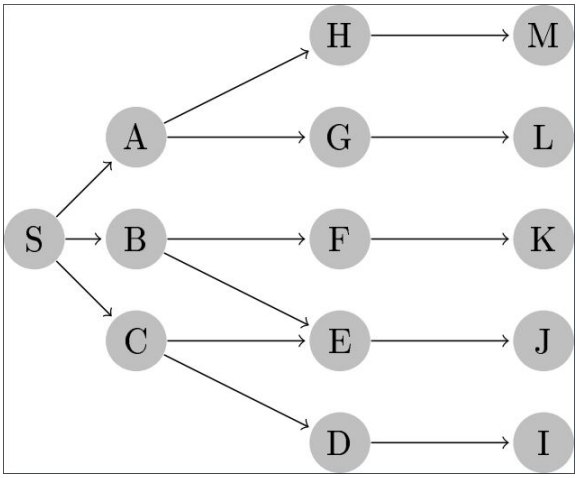
Solution: (a) is the correct answer. Being a bird entails being an animal, as the set of birds is a subset of the sets of animals.

19. (**AY 2019/2020 ST 1 Final Exam Question - Logic and Logical Agents**)
True/False: "Two agents with the same knowledge base and different inference engines, both of which are complete and sound, always behave in the same way".

In the rationale box: justify your answer.

Solution: False. The behavior of an agent describes its interaction with the environment. Different complete and sound algorithms will (given the same KB) result with the same inferences and a difference in how they derive that doesn't imply a difference in behavior. However, the behavior of an agent is not only dependent on its knowledge to different agents might have completely different objectives and hence act differently even with the same knowledge.

20. (**AY 2019/2020 ST 1 Final Exam Question - Inference**)
Consider the following AND-OR graph, where S = `True` and J is the query.



Assuming that ties are broken based on alphabetical ordering (i.e., when considering both A and B simultaneously, A is considered before B), state which is more efficient: Forward Chaining or Backward Chaining. Justify your answer.

Solution: False. Backward chaining is more efficient. Backward chaining would only explore J, E, B, C, and S. Forward chaining would explore many more nodes A,B,C, H, G, ...

21. (**AY 2019/2020 ST 1 Final Exam Question - Inference**)
Given the Knowledge Base (KB):

- $\neg((a \wedge \neg b) \vee (b \wedge c))$
- $\neg e \Rightarrow a$

And the statement, α

- $e \vee \neg c$

Use the Resolution Algorithm to determine if we may infer α from KB. Show your complete working for this resolution. You should make use of any relevant logical symbols $\neg, \wedge, \vee, \Rightarrow$ in your answer, or directly substitute them with the English words, not, and, or, implies respectively, with brackets where necessary.

Solution:

(a) $\neg((a \wedge \neg b) \vee (b \wedge c)) = \neg(a \wedge \neg b) \wedge \neg(b \wedge c)$
  (i) $\neg(a \wedge \neg b) = \neg a \vee b$
  (ii) $\neg(b \wedge c) = \neg b \vee \neg c$

(b) $\neg e \Rightarrow a = e \vee a$

(c) $\neg(e \vee \neg c) = \neg e \wedge c$
  (i) $\neg e$
  (ii) c

(d) $R(b) + R(c)(i) = a$

(e) $R(a)(i) + R(d) = b$

(f) $R(a)(ii) + R(e) = \neg c$

(g) $R(c)(ii) + R(f) = \emptyset$ (done)