1 **(a)** **(i)** T

(ii) T

(iii) T

(iv) F

(v) F [It is possible for DFS space complexity (bd) to be bigger than BFS (b^d). For example, when b = 1 and d = 5, DFS space complexity = bd = 5b, BFS space complexity = b^d = b]

(b) Yes, suppose h_2 = exact distance from goal while h_1 = 0, then its easy to see that the search using h_1 degrades to Dijkstra which will expand more nodes than the nodes on the path while h_2 is most optimal and expands only nodes on the path.

(c) Constraint propagation is the propagation of implications of a constraint on one variable onto other variables, whereby nodes are only expanded if they do not violate the constraint. This includes the use of most constraining variable, which helps to reduce the branching factor on future choices by selecting the variable that is involved in the largest number of constraints on unassigned variables, thus speeding up the search process.

Forward checking propagates information from assigned to unassigned variables, keeps track of remaining legal values for unassigned variables and terminates search when any variable has no legal values. This helps to prevent unnecessary expansion of nodes when the node has no more possible values to take on and speeds up the search process.

(d) Minimax is a search strategy that helps to find a sequence of moves that leads to a terminal state (goal). In Minimax the two players are called maximizer (MAX) and minimizer (MIN). MAX tries to get the highest utility possible while MIN tries to get the lowest utility possible. As the maximizer player and assuming that the opponent will play optimally, the complete search tree is generated. As a backtracking based algorithm, all possible moves are tried, then backtracked and a decision is made.

Minimax is a 3-step process:

1. Generate the entire game tree down to terminal states.
2. Calculate utility
   a) Assess the utility of each terminal state
   b) Determine the best utility of the parents of the terminal state
   c) Repeat the process for their parents until the root is reached
3. Select the best move, i.e. the move with the highest utility move

2 **(a)**

$$V_{i+1}(s) = \max_{\alpha} Q_{i+1}(s,a)$$

Based on the current estimates of Q(s,a) and assuming that the V value for State 6 (terminal state) is 0, the V values are as such:

V(1) = 4	V(2) = 8	V(3) = 9	V(4) = 5	V(5) = 8	V(6) = 0

Using the current V and Q values,

$$Q_{i+1}(s,a) = \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_i(s')]$$

$Q(1, \text{up}) = 1[(-1) + (0.9)5] = 3.5$			$Q(1, \text{right}) = 1[(-1) + (0.9)8] = 6.2$
$Q(2, \text{up}) = 1[(-1) + (0.9)8] = 6.2$		$Q(2, \text{left}) = 1[(-1) + (0.9)4] = 2.6$	$Q(2, \text{right}) = 1[(-1) + (0.9)9] = 7.1$
$Q(3, \text{up}) = 1[(10) + (0.9)0] = 10$		$Q(3, \text{left}) = 1[(-1) + (0.9)8] = 6.2$	
	$Q(4, \text{down}) = 1[(-1) + (0.9)4] = 2.6$		$Q(4, \text{right}) = 1[(-1) + (0.9)8] = 6.2$
	$Q(5, \text{down}) = 1[(-1) + (0.9)8] = 6.2$	$Q(5, \text{left}) = 1[(-1) + (0.9)5] = 3.5$	$Q(5, \text{right}) = 1[(10) + (0.9)0] = 10$

The updated V values are:

V(1) = 6.2	V(2) = 7.1	V(3) = 10	V(4) = 6.2	V(5) = 10

(b)

$$Q_{new}(S_t, A_t) \leftarrow Q_{old}(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{\alpha} Q_{old}(S_{t+1}, a) - Q_{old}(S_t, A_t))$$
$$Q(2, \text{up}) = 6 + 0.2(-1 + 0.9(8) - 6) = 6.04$$
$$Q(5, \text{right}) = 8 + 0.2(10 + 0.9(0) - 8) = 8.4$$

(c) Monte Carlo approach requires the entire trajectory which involves waiting until the end of the episode to determine the value V(s). It makes use of all the trajectories for taking an action a at state s (s,a) and average them to get the expectation.

Q-Learning only needs one-step trajectory to estimate the value of Q(s,a), which is defined by the action value of taking the action a at the state s. It uses the temporal difference update rule to estimate the value of Q at every time step the agent interacts with the environment.

(d) Monte Carlo approach involves taking the average of returns for state-action pair (s,a) visited in an episode, before choosing the optimal action based on the maximal return of all Q(s,a), as illustrated in the steps from the MC control algorithm below:

$$Q(s,a) \leftarrow \text{average}\big(\text{Returns}(s,a)\big)$$
$$a^* \leftarrow \text{argmax}_a Q(s,a)$$

3 **(a)**
(i) T
(ii) F
(iii) T
(iv) F
(v) F

(b) **(i)** To prove the forward direction:

$$A \land B \land C \Rightarrow D \equiv \neg(A \land B \land C) \lor D$$
$$\equiv \neg A \lor \neg(B \land C) \lor D$$
$$\equiv \neg A \lor \neg B \lor \neg C \lor D$$

To prove the reverse direction:

$$\neg A \lor \neg B \lor \neg C \lor D \equiv \neg(A \land B) \lor \neg C \lor D$$
$$\equiv \neg(A \land B \land C) \lor D$$
$$\equiv A \land B \land C \Rightarrow D$$

From the above equivalences, the statement is valid.

(ii)

A	B	C	D	¬A	¬B	¬C	A∧B∧C	A∧B∧C⇒D	¬A∨¬B∨¬C∨D
T	T	T	T	F	F	F	T	T	T
T	T	T	F	F	F	F	T	F	F
T	T	F	T	F	F	T	F	T	T
T	T	F	F	F	F	T	F	T	T
T	F	T	T	F	T	F	F	T	T
T	F	T	F	F	T	F	F	T	T
T	F	F	T	F	T	T	F	T	T
T	F	F	F	F	T	T	F	T	T
F	T	T	T	T	F	F	F	T	T
F	T	T	F	T	F	F	F	T	T
F	T	F	T	T	F	T	F	T	T
F	T	F	F	T	F	T	F	T	T
F	F	T	T	T	T	F	F	T	T
F	F	T	F	T	T	F	F	T	T
F	F	F	T	T	T	T	F	T	T
F	F	F	F	T	T	T	F	T	T

From the truth table, the two propositions are equivalent.

(c) **(i)** All footballers are busy on some weekends.

(ii) Every person loves another person.

(d)

∀x, Athlete(x)∧Fit(x)∧Smart(x)∧Lucky(x) ⇒ Gold_medal(x) (1)
∀x, Train_hard(x) ⇒ Fit(x) (2)
∀x, Clever(x) ⇒ Smart(x) (3)
Lucky(Don) (4)
Train_hard(Don) (5)
Clever(Don) (6)
Athlete(Don) (7)

From (3) and Universal-Elimination:

Clever(Don) ⇒ Smart(Don) (8)

From (6), (8) and Modus Ponens:

Smart(Don) (9)

From (2) and Universal-Elimination:

$$\text{Train\_hard(Don)} \Rightarrow \text{Fit(Don)} \quad (10)$$

From (5), (10) and Modus Ponens:

$$\text{Fit(Don)} \quad (11)$$

From (1) and Universal-Elimination:

$$\text{Athlete(Don)} \wedge \text{Fit(Don)} \wedge \text{Smart(Don)} \wedge \text{Lucky(Don)} \Rightarrow \text{Gold\_medal(Don)} \quad (12)$$

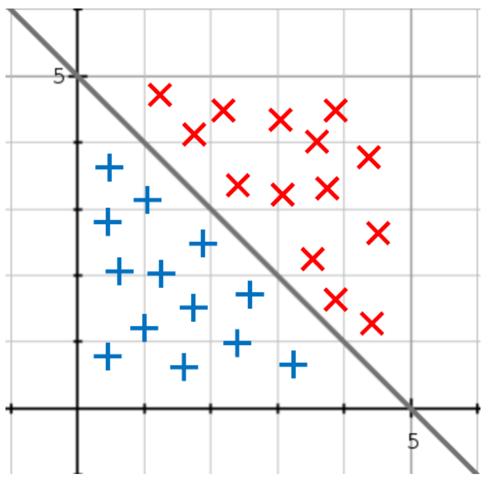
From (4), (7), (9), (11) and And-Introduction:

$$\text{Athlete(Don)} \wedge \text{Fit(Don)} \wedge \text{Smart(Don)} \wedge \text{Lucky(Don)} \quad (13)$$

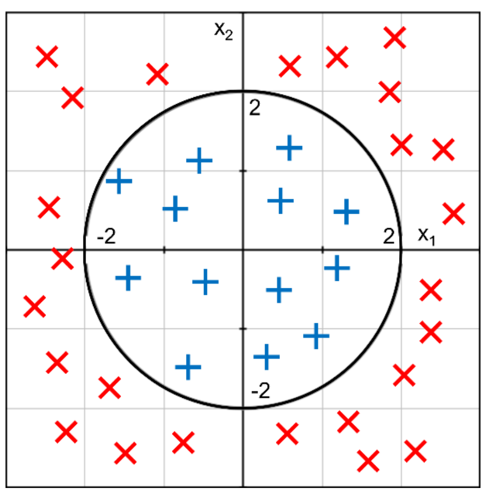
From (12), (13) and Modus Ponens:

$$\textbf{Gold\_medal(Don)} \quad (14)$$

4 (a) (i)



(ii)



(b) (i)

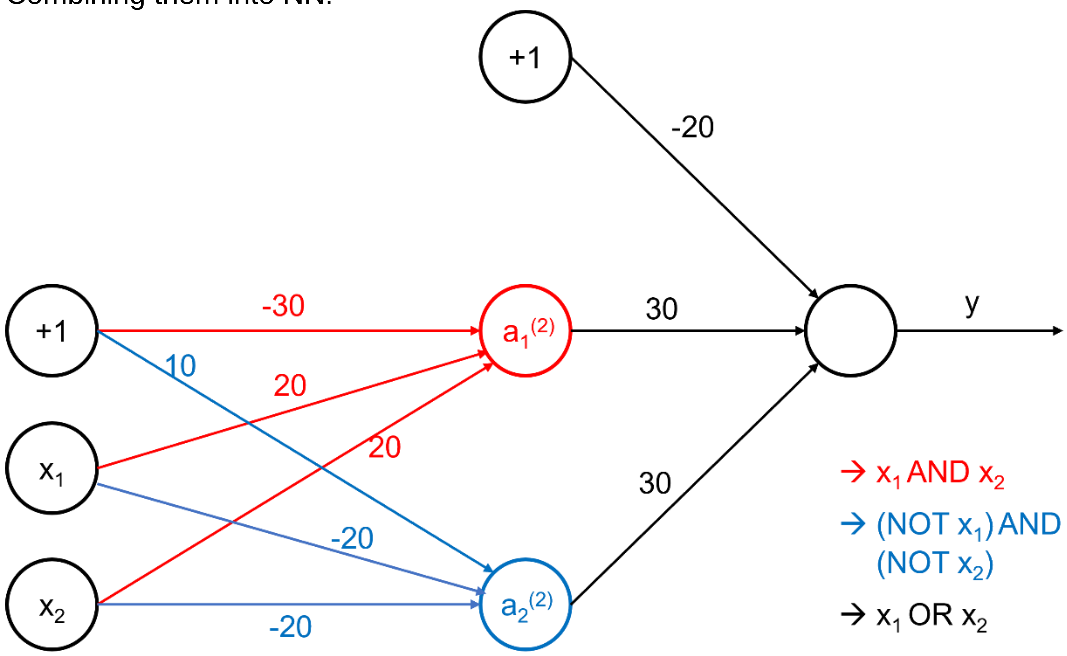
x_1	x_2	y
0	0	$g(-20) \approx 0$
0	1	$g(10) \approx 1$
1	0	$g(10) \approx 1$

1	1	$g(40) \approx 1$

(ii) OR

(iii) To implement XNOR function (aka NOT XOR), positive output is produced if and only if
- AND (both true)
- Neither

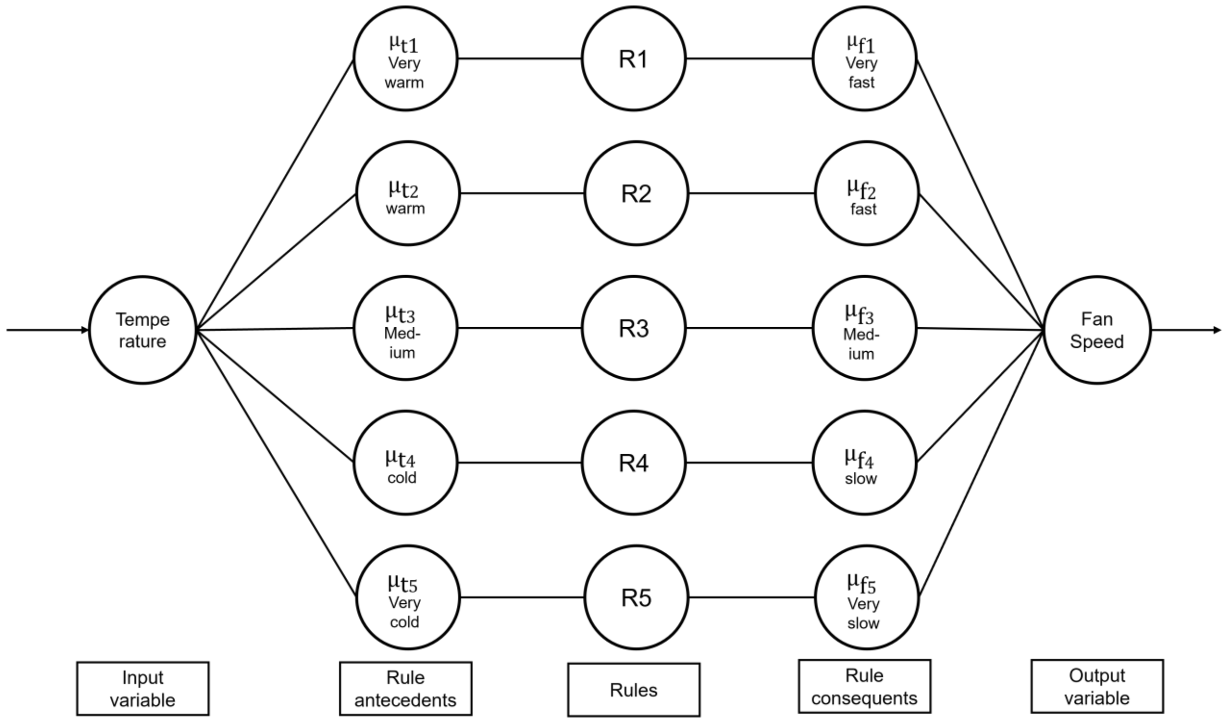
Combining them into NN:



Truth table:

x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	**y**
0	0	0	1	**1**
0	1	0	0	**0**
1	0	0	0	**0**
1	1	1	0	**1**

(c)



Solver: Lin Lejia (LINL0031@e.ntu.edu.sg)