All questions were True/False.

1. There were two possible questions:
   - Imagine you have a search problem where you have a perfect heuristic function \( h(s) \) - in other words, for all states \( s \), \( h(s) = h^*(s) \), the minimum cost to get from \( s \) to a goal state. If you use hill climbing with \( f(s) = h(s) \) using this heuristic function, hill climbing is guaranteed to find an optimal solution. (Assume lower values are better for hill climbing.)
     
     **False.** \( h(s) \) considers just the cost from a state to the solution and not the cost of the path to get to the state. Even though the lower-cost solution is the goal three steps away on the left, hill climbing would choose to go right from the root, yielding a solution of cost 11 rather than 3.

   - Imagine you have a finite search space whose branching factor is 1 for all states. Hill climbing might fail to find a solution if one exists.
     
     **True.** It depends on the function that you use. Since you only go to a successor if the best successor has a value that’s better than the current state, if you get to a state whose sole successor is worse, the search would stop even if you didn’t reach a solution.

2. There were two possible questions:
   - If \( h(s) \) is admissible, then \( \max(0, \log_2 h(s)) \) is admissible.
     
     **True.** Since \( h(s) \) is admissible we know that \( 0 \leq h(s) \leq h^*(s) \). Let’s use \( h'(s) = \max(0, \log_2 h(s)) \). To see that \( h'(s) \) is admissible we need to verify that \( 0 \leq h'(s) \leq h^*(s) \). Let’s consider separately the two cases for the max – which is larger depends on whether \( h(s) \) is above or below 1. If \( h(s) \leq 1 \) then \( \log_2 h(s) \leq 0 \), so \( h'(s) = \max(0, \log_2 h(s)) = 0 \), which trivially satisfies admissibility. If \( h(s) > 1 \), recall that \( \log_2 x < x \) (that is, the curve for \( y = \log_2 x \) is always below the curve for \( y = x \)), so \( h'(s) = \max(0, \log_2 h(s)) = \log_2 h(s) < h(s) \leq h^*(s) \), plus since \( h(s) > 1 \) we know \( h'(s) = \max(0, \log_2 h(s)) = \log_2 h(s) > 0 \). So regardless of whether \( h(s) \) is above or below 1 we find that \( h'(s) \) is admissible.

   - If \( h(s) \) is admissible, then \( \sqrt{h(s)} \) is admissible.
False. As one example, imagine you have a problem where all edge costs are 0.1. If the nearest solution is less than 10 steps from the initial state then $h^*(s) < 1$. Therefore since $h(s)$ is admissible it too must be less than 1. Recall that square root has the property that if $0 < x < 1$ then $x < \sqrt{x}$. So if $h(s)$ is less than 1 its square root is larger than $h(s)$, and could yield a result that exceeds $h^*(s)$.

3. Consider a search problem where all actions have cost greater than or equal to 1, and you have a heuristic function for which for all states $s \leq h(s) \leq 1$ and if $s$ is a goal state $h(s) = 0$. A* will find an optimal solution using this function.

   True. If you’re at a goal state $h(s)$ is correctly 0. At all other states $h^*(s) \geq 1$ because there’s at least one edge between it and the nearest goal and all edge costs are 1. Thus we have $0 \leq h(s) \leq 1 \leq h^*(s)$, meaning $h(s)$ is admissible so A* will find an optimal solution.

4. Imagine you have a search space (potentially a tree or a graph) that has a total of n states. Beam search with a beam size equal to $n$ is guaranteed to find a solution. (Assume the search method checks for cycles in the usual fashion, using the Closed list.)

   True. The beam size is how much of the Open list you keep after each iteration of expanding a state. If there are $n$ states then Open can’t be bigger than $n$, and beam search doesn’t need to throw anything away if that’s the beam size. Eventually beam search will run through all the states until it finds a goal state (it need not be an optimal solution, just a solution).