1.

a. **What does it mean for a heuristic to be admissible?**

A heuristic is admissible when it always underestimates the true cost to reach the goal.

b. **In class, we discussed two different heuristics for the 15-puzzle. The first, h1, was the number of misplaced tiles. The second, h2, was the manhattan distance between each tile and its correct location.**

i. **Are h1 and h2 admissible? Please prove your answer.**

Both h1 and h2 are admissible. For h1, this is because each misplaced tile must move at least once to reach its correct position, so the number of misplaced tiles cannot overestimate the true cost. For h2, this is because each tile must move at least its Manhattan distance to reach its goal position, and tiles cannot move diagonally or skip over spaces.

ii. **Which heuristic is preferred? Please prove your answer.**

h2 (Manhattan distance) is preferred because it provides a larger estimate than h1 while still being admissible, meaning it dominates h1. This proves that h2 provides a closer heuristic to the true cost.

2.

6. **Run each of the four algorithms (breadth_first_search, depth_first_search, depth_limited_search, iterative deepening search) on the RoverState and count the number of states generated.**

   Please add to your written answers a table with the state data for each of the questions above.

	Breadth-First Search	**Depth-First Search**	**Depth Limited Search (limit=8)**	**Iterative Deepening Search**
Number of States Generated	127 states	59 states	49 states	344 states

3.

d. **Run both A* and uniform cost search (i.e. using h1: h=0 for all states) on the MarsMap and count the number of states generated. Add this to your results.**

	A*	**Uniform Cost Search**
Number of States Generated	42 states	62 states

4. **Run the included code in nurse_scheduler.py. Read through it and see if you can understand what it does. Then follow the instructions below and add a section to your answers explaining what you learned.**
    a. **Increase the number of nurses to 7, and add shift requests for them. Does this make the problem easier or harder? How can you tell?**

Increasing the number of nurses from 5 to 7 made the problem easier to solve, as shown by the statistics in the output. First, the number of conflicts decreased significantly from 26 to 2, indicating fewer constraint violations needed to be resolved. Second, the solver was able to meet 19 out of 21 shift requests (90.5%) compared to the original 13 out of 20 (65%), showing that having more nurses available made it easier to accommodate individual scheduling preferences.

    b. **Decrease the number of nurses to 3 and remove the extra shift requests. Does this make the problem easier or harder? How can you tell?**

When the number of nurses was reduced to 3, the problem became significantly harder to solve. First, the number of conflicts increased significantly from 26 and 2 to 93, indicating the solver had to work harder to find a valid solution. The solution quality also decreased, with only 10 out of 21 shift requests (47.6%) being met compared to the previous 65% and 90.5%. This decline occurred because having fewer nurses meant less flexibility in meeting individual scheduling preferences while still satisfying the basic constraint that every shift must be covered.

c. **Return to five nurses, but increase the number of shifts to five, and update the shift requests. Does this make the problem easier or harder? How can you tell?**

Increasing the number of shifts from 3 to 5 made the problem impossible to solve, as indicated by the "No optimal solution found" message. This is a significant change from the previous scenarios where solutions were found. This makes sense because with 5 shifts per day and only 5 nurses, while maintaining the constraint that each nurse works at most one shift per day, there aren't enough nurses to cover all shifts while satisfying the minimum shifts per nurse requirement.

d. **In this problem, we've separated the problem knowledge from the algorithmic knowledge. How does this help us in engineering a solution?**

Separating problem knowledge from algorithmic knowledge helps engineer solutions by allowing us to focus on defining constraints that guide the search process. Problem knowledge, such as scheduling rules and shift requests, can be used to control how the search tree expands. For example, infeasible solutions, like schedules that have nurses working multiple shifts a day, can be eliminated early in the search process. This separation lets us modify our constraints without changing the solver's logic, while ensuring the search focuses only on potentially valid solutions, making the optimization process more efficient.

5. **Deep Blue vs AlphaZero (10 points) (Feb 27)**

    a. **What were the engineering advances that led to Deep Blue's success? Which of them can be transferred to other problems, and which are specific to chess?**

Deep Blue achieved its success through a combination of hardware and software innovations. On the hardware side, it used a massively parallel system and custom-designed chess chips that could rapidly evaluate positions and generate moves. On the software side, it used highly optimized alpha-beta pruning with a handcrafted evaluation function. While the alpha-beta search algorithm and parallel processing concepts could be applied to other problems, much of Deep Blue's strength came from chess-specific optimizations. These included custom chips, evaluation functions with hand-tuned weights for pieces, and position-specific bonuses that were crafted by chess experts. This heavy reliance on domain expertise made the system highly effective at chess but difficult to adapt to other problems.

    b. **AlphaZero is compared to a number of modern game-playing programs, such as StockFish, which work similarly to Deep Blue. The paper shows that AlphaZero is able to defeat StockFish even when it is given only 1/100 of the computing time. Why is that? Please frame your answer in terms of search and the number of nodes evaluated.**

AlphaZero achieves superior performance despite evaluating far fewer position nodes because of how it guides its search. While Stockfish (like Deep Blue) uses alpha-beta search to evaluate around 70 million nodes per second, AlphaZero only evaluates about 80,000 nodes per second using Monte Carlo Tree Search. The key difference is that AlphaZero uses its neural network to

focus the search on promising moves, rather than having to explore many variations that a human expert would immediately recognize as poor. This more selective approach, which the paper describes as human-like, allows AlphaZero to make better use of each node it evaluates. This demonstrates that intelligent search pruning can be more effective than brute-force calculation by examining far fewer nodes in the game tree.