2. Unlike Expectimax, Minimax 'can take a risk' and end up in a state with a higher utility as opponents are random. [1 point]
False, based on the nature of expectimax and minimax search algorithms.

3. The importance of an accurate evaluation function increases by increasing the depth. [1 point]

False, because we have more uncertainty further up in the tree; therefore, it is more important to have an accurate evaluation function higher up in the tree.

4. Identify the appropriate type of search for the problems listed below: [3 points]

CPU job scheduling -> goal itself is important
Graph coloring problem -> goal itself is important
Finding the shortest route from city A to city B -> path to goal itself is important
Chess game -> path to goal itself is important (This will be graded manually as the autograder response is incorrect.)

5. Specify <u>two problems</u> with the Hill-climbing algorithm and explain <u>two different methods (one for each problem)</u> that alleviate these problems.
[6 points - 3 points for two problems and 3 points for two reasonable solutions]

1. Problem 1
    a. We get stuck in a local optimum and not the global optimum.
    b. Solution: We can implement hill-climbing with random restarts, pick the best solution, or use the simulated annealing approach.
2. Problem 2
    a. Plateau points, a flat portion of the state-space landscape hill-climbing tries to find uphill parts, and if there are none, we are stuck.
    b. Solution: Implement a limited number of side-way movements in an attempt to leave the plateau.

6. How is local beam search equivalent to AND different from multiple parallel runs of hill-climbing algorithms? Explain these shared and dissimilar characteristics.
[4 points - 2 points for shared characteristics, 2 points for dissimilarities]

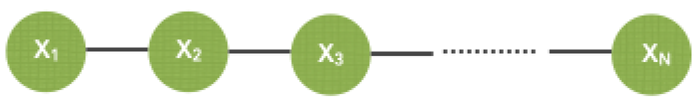
Similarities: If k=1, then the beam search is just a single thread of climbing a hill without keeping track of other good choices. They both consider k nodes at any given time.

Dissimilarities: Local beam search is not equivalent to parallel runs of several hill-climbing because, in the parallel runs, there is no sense of memory of the best states. Local beam search is like k parallel runs, but the information is shared, and successors are chosen by considering all neighbors.

7. Simulated annealing is more likely to accept a bad move late in the search than earlier. [1 point]

False, based on the definition of the temperature schedule in the simulated annealing algorithm.

8. Consider the general less-than chain CSP below. Each of the N variables X_i has the domain $\{1, 2, \ldots, M\}$. The constraints between adjacent variables X_i and X_{i+1} require that $X_i < X_{i+1}$. [10 points]



For now, assume $\mathrm{N} = \mathrm{M}$.

a. How many solutions does the CSP have? [2 points]

One solution: 1<2<3<.....<N

b. What will be the domain of X_1 after enforcing the consistency of only arc $X_1 \longrightarrow X_2$? [3 points]

X_1 domain = {1,2,3,...,N-1}

c. What will be the domain of X_1 after enforcing the consistency of only arc $X_2 \longrightarrow X_3$ and then $X_1 \longrightarrow X_2$? [3 points]

X_1 domain = {1,2,3…N-2}

d. What will be the domain of X_1 after fully enforcing arc consistency in the entire problem? [2 points]

X_1 domain = {1}

9. a. Why do we prefer to change nearly tree-structured CSPs to tree-structured CSPs? How would this conversion happen? [3 points]

We prefer tree-structured CSPs because this reduces the time complexity since the arc-consistency algorithm will not backtrack and consistency of arcs will be maintained as variables are assigned. O(nd^2) vs. O(d^n).

We can do this conversion by:
Identify the minimum number of nodes, that if removed, we are left with a tree (cut-set). Then we do cut-set conditioning (pick one of those nodes call it C and assign a value to them (conditioning), then remove conflicting values from the domain of the neighbors), and run the tree-structured algorithm on each instantiated problem.

b. Describe the algorithm for solving a tree-structured CSP and discuss its runtime.

[3 points]

Given a tree-structured CSP, we apply topological ordering (ordering nodes, from 1 to n, so that all parents come before their children in the sequence). Then:

i) Do a backward pass, from node n-1 down to 1, enforce arc consistency between the parent of node and the node itself (Parent(Xi) → Xi arc consistency)

ii) Do a forward pass selecting a value from the domain of each variable that is not conflicting with the values assigned previously.

Solving a tree-structured CSP gives us a quadratic runtime versus an exponential run time because now we have to check each node in the tree and at each node (O(n)), we perform arc consistency in O(d^2).

Runtime: O(nd^2)

10. If A and B are two events such that P(A) = 0.2, P(B) = 0.6 and P(A|B) = 0.2 then what is the value of P(~A |~B)? Show your work.
[3 points]

P(A|B) = 0.2 and P(B) = 0.6 so P(A, B) = 0.12 and therefore:
P(A, ~B) = P(A) - P(A, B) = 0.2 - 0.12 = 0.08

P(~A|~B) = 1- P(A|~B)
P(A|~B) = P(A, ~B)/P(~B) = 0.08/0.4 = 0.2 → P(~A|~B) = 1 - 0.2 = 0.8