Name: **1 pt**

Campus id: **1 pt**

0+1	2	3	4	5	6	total
2+60	12	30	30	26	30	**190**

UMBC CMSC 471 Midterm Exam 2023-10-25

Write your answers on this exam, which is open-note only and consists of six problems, summing to 190 points. You have the entire class period, seventy-five minutes, to work on the exam. There is a blank page at the end you can use for working out problems, please hand in with your exam. Good luck.

You are only allowed the following materials as part of the open note policy:

- Lecture slides
- Reference Book (hardcopy or digital copy)
- Your notes (handwritten or digital device)

____
Signature

No other print outs or digital copies should be opened during the exam, including the practice exams, quizzes, or homework. You are responsible for following the honor code.

1. True/False [60 points; 3 pts each]

Circle **T** or **F** in the space before each statement to indicate whether the statement is **true** or **false**.

T F In partially-observable, deterministic environments, agents need not deal with uncertainty.

T F A greedy algorithm makes locally optimal choices in hope of finding a global optimum.

T F Depth-first search will always expand more nodes than breadth-first search.

T F There is generally no difference in the performance of a **depth-first graph search** and a **breadth-first graph search** algorithm for a problem with a finite state-space graph.

T F If a solution exists and all actions have unit cost, the **iterative deepening** search approach may sometimes fail to find a solution.

T F A **depth-first tree-search or graph-search** has an advantage over breadth-first searches in that it will eventually find a solution if one exists.

T F A **local search** algorithm typically has the advantage of needing less memory than a tree-search or graph-search algorithm.

T F The **hill climbing** search algorithms cannot be used to solve problems that have more than two dimensions.

T F The heuristic function **h(n) = 0** is admissible for every search problem.

T F The **min-conflicts local search** technique for solving a search problem will always find a solution, given enough time.

T F In a constraint satisfaction problem's **constraint graph**, each node represents a variable, and each arc represents a constraint.

T F A CSP constraint graph should not contain any loops.

T F When enforcing arc consistency in a CSP, the set of values which remain when the algorithm terminates does not depend on the order in which arcs are processed from the queue.

T F The **minimum remaining values algorithm** for solving a constraint satisfaction problem chooses the variable with the fewest legal values in the remaining variables.

T F In constraint satisfaction problems, variables must have a finite set of possible values.

T F The **minimax algorithm** can only be used for zero-sum games.

T F When using alpha-beta pruning, it is possible to get an incorrect value at the root node by choosing a bad ordering when expanding children.

T F When using alpha-beta pruning, the computational savings are independent of the order in which children are expanded.

T F Chess, checkers and go are examples of games that have a partially observable environment. FALSE. The entire state is visible in the board for these games.

T F The minimax and alpha-beta procedures will always back up identical values to the root node of a game tree for any possible static evaluation function.

2. Admissibility True/False (12 points)

Circle **T** or **F** in the space before each statement to indicate whether the statement is **true** or **false**.

2.1 Assume that **h1(n)** and **h2(n)** are **both admissible** heuristics for an algorithm A search problem. Answer the following True/False questions. (6 points, 2 each)

T F The heuristic **max(h1(n), h2(n))** is necessarily admissible.

T F The heuristic **min(h1(n), h2(n))** is necessarily admissible.

T F The heuristic **(h1(n) + h2(n))/2** is necessarily admissible.

2.2 Assume that h1(n) is an **admissible** heuristic, and that h2(n) is an **inadmissible** heuristic. Answer the following True/False questions. (6 points, 2 each)

T F The heuristic **max(h1(n), h2(n))** is necessarily admissible.

T F The heuristic **min(h2(n), h2(n))** is necessarily admissible.

T F The heuristic **(h1(n) + h2(n))/2** is necessarily admissible.

3. Constraint Satisfaction Problems [30 points]

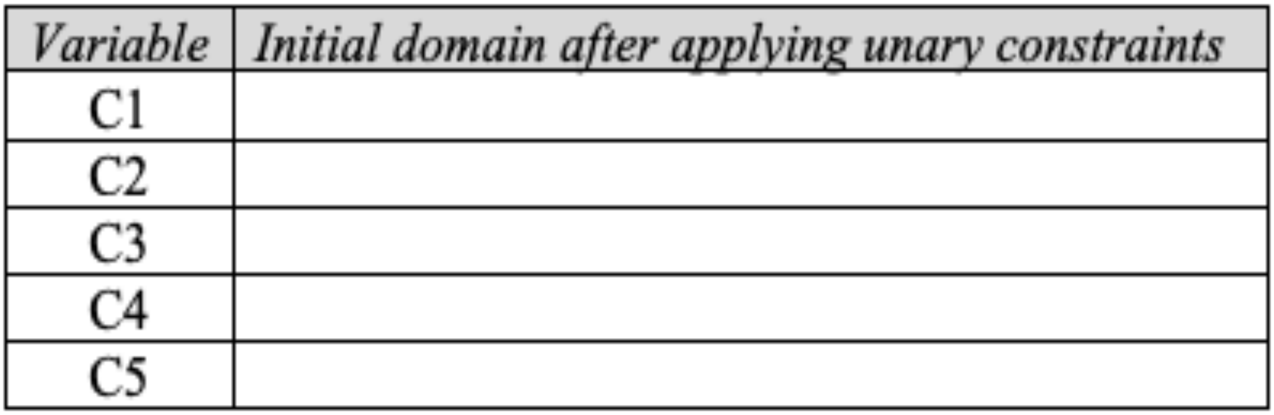
There are five MWF graduate CS classes and three instructors who will be teaching these classes. An instructor can only teach one class at a time. The classes are:

- Class 1 - Algorithms: meets from 8:00-9:00am
- Class 2 - Intro to AI: meets from 8:30-9:30am
- Class 3 - Databases: meets from 9:00-10:00am
- Class 4 - Operating Systems: meets from 9:00-10:00am
- Class 5 - Machine Learning: meets from 9:30-10:30am

and the instructors are:

- Professor A, who is available to teach Classes 3 and 4.
- Professor B, who is available to teach Classes 2, 3, 4, and 5.
- Professor C, who is available to teach Classes 1, 2, 3, 4, 5.

(a) [8] Formulate this as a CSP problem in which there is one variable per class. State the variables and their domains, and the constraints. Constraints should be unary or binary and specified formally and precisely, but may be implicit rather than explicit.



Variable	*Initial domain after applying unary constraints*
C1	
C2	
C3	
C4	
C5	

(b) [7] Draw the constraint graph associated with your CSP.

(c) [8] Show the domains of the variables after running arc-consistency on this initial graph (after having already enforced any unary constraints).

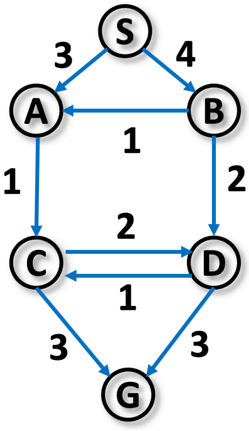
(d) [7] Show each of the possible solutions.

4. Search [30 points]

Consider this **problem-space graph**, where S is the start node, G is the goal node, and the number next to each edge is the cost to go from the state at its tail to the state at its head.

The table shows the values of a **heuristic function h** for each state, which estimates the cost of the best path from that state to a goal. We do not know if h is admissible or not. The **h*(n) column** will hold the true minimal cost of a path from node n to the goal node G.

Now, we run **algorithm A** on this graph using the heuristic function **h(n)**, whose values are given in the table.



n	h(n)	h*(n)	h1(n)
S	5	7	4
A	3	4	3
B	1	5	2
C	1	3	1
D	2	3	1
G	0	0	0

4.1 Enter a *string* that is the sequence of the **states expanded** (not just noticed and added to the graph) by the algorithm in the order that they are expanded, e.g., *SACG*. In choosing which node to expand next, assume ties are broken by selecting the one whose name appears first in an alphabetic ordering. [5 pts]

4.2 Enter a *string* which is the sequence of the states that represent the final **solution path found** by algorithm A using the given heuristic h(n) from S to G, e.g., *SBDG*. [5 pts]

4.3 Enter a *number* that is the **cost of the path** found, e.g., *9*. [5 pts]

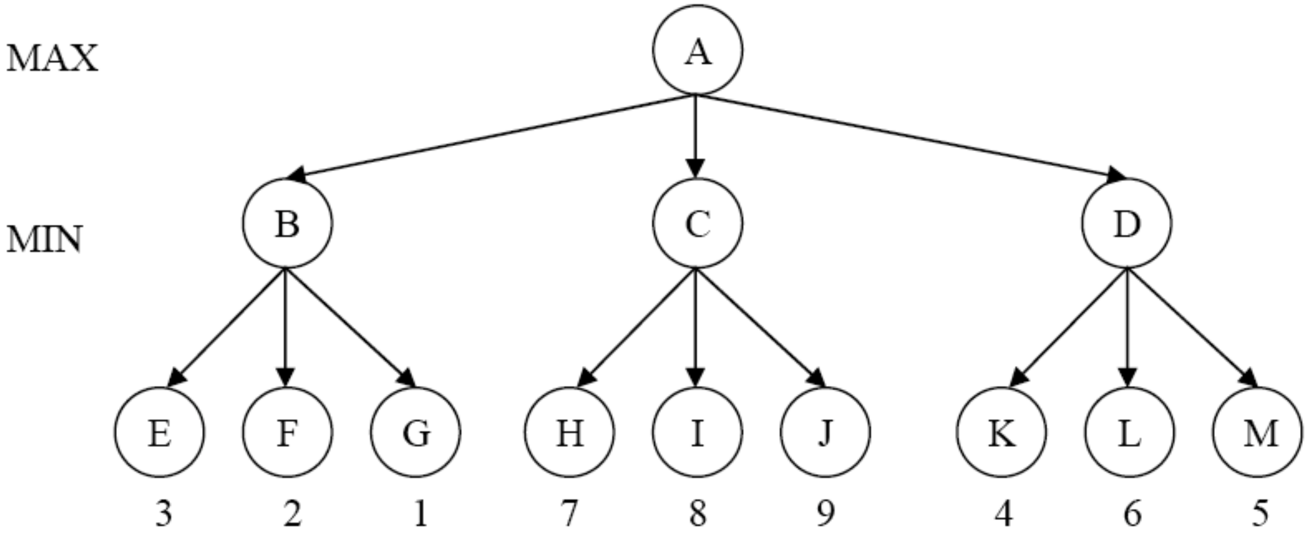
4.4 Is the heuristic function h **admissible**? Enter *yes* or *no*. [5 pts]

4.5 Is the heuristic function h1 **admissible**? Enter *yes* or *no*. [5 pts]

4.6 Which heuristic is better – h or h1? Why? [5 pts]

5. Game Trees [26]

Consider this game tree where the root is a maximizing node, and children are visited left to right.



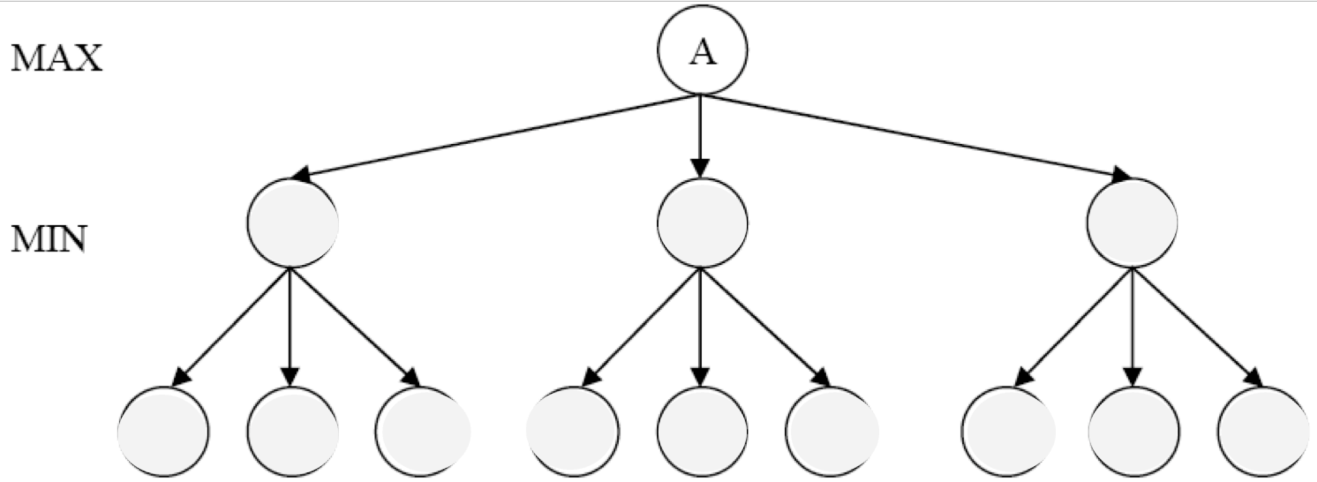
(a) [5] Compute the minimax game value of nodes A, B, C, and D using the standard algorithm.

A =   ; B =   ; C =   ; D =

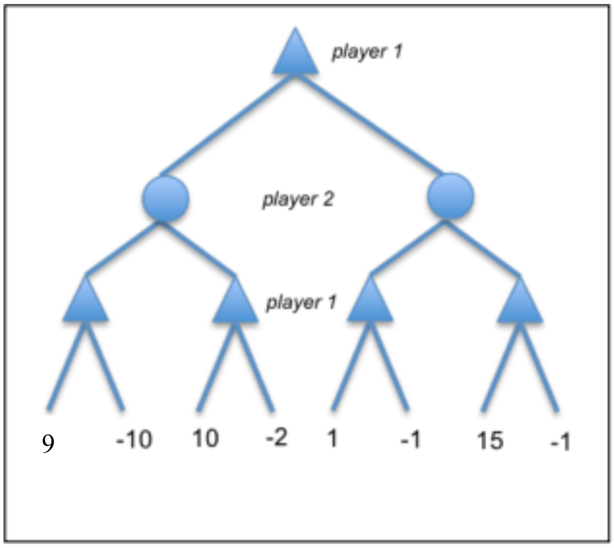
(b) [5] What move will be selected by player one using minimax.

(c) [5] List the nodes (leaves or interior) that alpha-beta prunes, i.e., decides need not be examined.

(c) [5] The number of nodes that alpha-beta prunes depends on how each node's children are ordered in the tree. Draw a new game tree by re-ordering the children of each internal node, such that the new game tree is equivalent to the tree above, but alpha-beta pruning will prune as many nodes as possible. List the nodes that would be pruned.



(e) [6] Assume that player 2 chooses an action uniformly at random every turn and that player 1 knows this. Player 1 still seeks to maximize her payoff. Circle player 1's optimal next move, and give her expected payoff. Show the expected value at each node in the tree.



6. Short answer (30 pts)

6.1 Imagine you are an AI game developer working on a strategy game where Monte Carlo Tree Search (MCTS) is employed to make decisions for an AI player.

(a) Why do you think you would need MCTS instead of MiniMax algorithm? (2 pts)

(b) How does MCTS expand the search tree compared to MiniMax model? (2 pts)

(c) Explain the process of adding new successor in the search tree in few simple steps. (6 pts)

6.2 What is a main advantage of A* search over hill-climbing search? (6 pts)

6.3 Which search algorithm uses randomness to avoiding getting trapped in local maxima? (2 pts)

6.4 A search algorithm is complete if it… (2 pts)

a. Always finds the optimal solution

b. Finds all possible solutions

c. Never finds a solution

d. Always finds a solution if there is one

6.5 In the problem solving as search paradigm, explain in a few sentences why it does or does not make sense to allow an **action whose result is the same state as the one it is performed in**. Mention issues for both tree-based and graph-based search algorithms. (10 pts)