Name:
Student ID:

CM 146
Midterm Exam

This is a take-home exam, **due Tuesday, 2/18/25 at 11:59 PM**. It is open notes & materials.

You should submit it electronically via the submission link in the assignment. Please submit a single pdf file with your answers to the problems in order and with problem numbers labeled. Remember to include your name in the file.

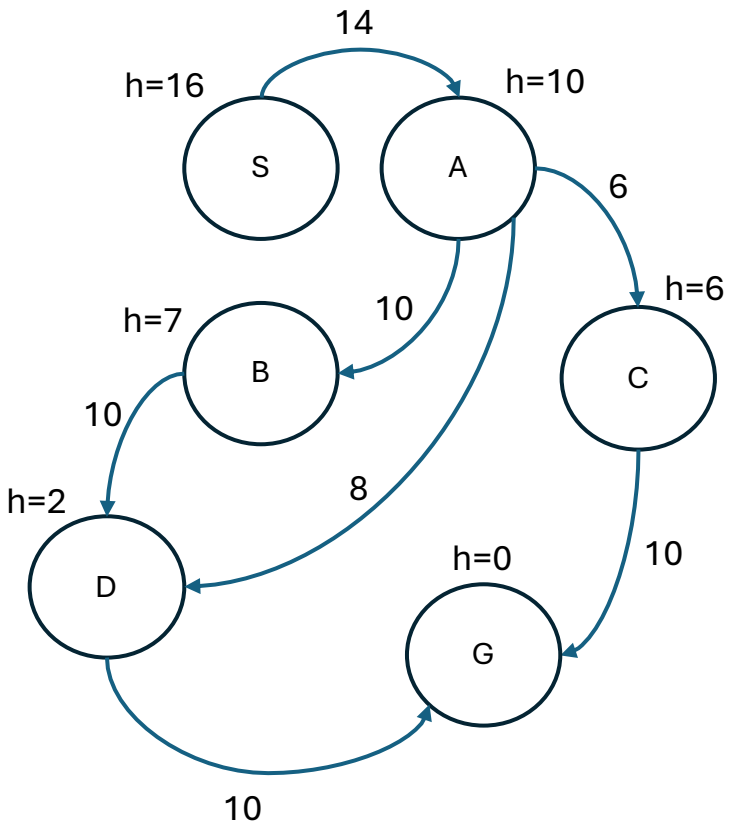
As usual, this exam must represent your personal work, per university policy. Please do not discuss the exam with other students until after the midterm is due.

All Q&A about the exam must be on Canvas (vs Discord). We will monitor the Canvas discussion and try to answer questions as soon as possible.

You may post clarification questions but not answer them for other students.

CM 146, Midterm
20 points

1. Consider the following directed graph where S is the start and G is the goal, transition costs are on arcs, and h is an admissible heuristic (giving a lower bound on the distance to the goal).



A. **6 points**. Apply A* to find a path from S to G. Fill out the following table, showing the full contents of the priority queue at the start of each step (before a node is popped from the queue). Show the final path and cost in the last line of the table. Write your entries in this form:

(<node name>, <priority>, <node name>←<parent node name>) *

Here, * means 0 or more entries of this form. For example, the following entry shows a queue with two nodes, W and K, at priorities 20 and 25, both found by traversing a link from J.

(W, 20, W←J) (K, 25, K←J)

B. **3 points:** What path and cost does Greedy Best First Search find?

C. **3 points**. What path and cost do Dijkstra's algorithm find?

D. **2 points.** Explain why the paths found in parts A and B are the same, or different.

E. **2 points.** Explain why the paths found in parts B and C are the same, or different.

F. **2 points.** List the algorithms (Dijkstra, Greedy Best First Search, and A*) in ascending order of the number of nodes they expand (generate successors for) in this problem. Identify how many nodes each algorithm expands.

G. **2 points.** Explain why A* and Dijkstra explore the same, or a different number of nodes.

CM 146, Midterm
20 points

2. Watch the great chomper scene from Galaxy Quest (click on the image):

If anyone were so cruel as to build the chompers shown in this movie, they would also need a method of cleaning up after any mess the chompers would sadly produce. Here, we delegate that task to a cleaning robot (not seen in Galaxy Quest) that is controlled by the same mechanism that operates the chompers.

Your task is to compose a single Hierarchical Finite State Machine that controls a system composed of two chompers, like those in the movie, together with the cleaning robot.

Your HFSM can employ the following events:

- Heartbeat Detected (H)
- No Heartbeat Detected (NH)
- Mess (M)
- No Mess (NM)
- nSec – n seconds have elapsed since entering a state

Your HFSM should have the following behaviors:

- The chompers are active if and only if the robot is idle.
- The robot is cleaning if and only if the chompers are idle.
- Idle chompers are open.

- An idle robot is hiding.
- Active chompers begin in an armed state, waiting for anything with a heartbeat to arrive.
- They return to the armed state if the lifeform (or lifeforms) with the heartbeat escapes.
- Once chomping, chompers chomp until nothing with a heartbeat remains.
- The robot is only active when there is a mess.
- The robot returns to idle when the mess is gone, but it never stays active for more than 4 minutes.
- The robot cleans by sweeping the floor, hosing down the equipment, blasting everything with a flamethrower, and then polishing out any remaining spots, each for 30 seconds. It repeats this process indefinitely, until it is told to stop.
- Each chomper follows a regular, timed pattern where it opens, pauses, and closes. Each chomper takes a different amount of time to complete its pattern.
- When the chompers start chomping, Chomper A's first chomp is immediate. Chomper B's first chomp is 1 second later.
- The system begins with the chompers active

The following questions take you through pieces of the problem. When you are asked to provide diagrams, submit a drawing of an HFSM that clearly labels the states, transitions, transition events, and actions that take place in each state (if they are not synonymous with the state's name).

(a) 2 pts. Name the states, actions, and events in this component task:

- The robot cleans by sweeping the floor, hosing down the equipment, blasting everything with a flamethrower, and then polishing out any remaining spots, each for 30 seconds. It repeats this process indefinitely, until it is told to stop.

For question (a), assume the robot enters a state called DONE when it stops cleaning.

(b) 2 pts. Draw an HFSM that meets the specification in (a). Label it "bot cleaning".

(c) 4 pts. Draw an HFSM that meets the following specification for chomper behaviors. Label it "Chomping".

- Each chomper follows a regular, timed pattern where it opens, pauses, and closes. Each chomper takes a different amount of time to complete its pattern.
- When the chompers start chomping, Chomper A's first chomp is immediate. Chomper B's first chomp is 1 second later.

(d) 2 pts. Draw an HFSM that adds the following behavior to your solution for (c). Label it "Chompers active".

- Active chompers begin in an armed state, waiting for anything with a heartbeat to arrive.
- They return to the armed state if the lifeform (or lifeforms) with the heartbeat escapes.
- Once chomping, chompers chomp until nothing with a heartbeat remains.

(e) 5 pts. Draw an HFSM that implements the high-level structure required by the following specifications. Include the names of the top-level hierarchical states, but do not show their substructure. Draw arrows for the transitions, but do not label the conditions.

- The chompers are active if and only if the robot is idle.
- The robot is cleaning if and only if the chompers are idle.

(f) 5 pts. Draw an HFSM that integrates your component solutions for (a) – (e) into a single system that meets the full specification. Show all states and transitions. Make sure the conditions governing transitions are clear and unambiguous.

CM 146, Midterm
20 points

3. The behavior trees presented in class have encoded deterministic (if time variant) mappings from situation to action. However, characters often seem smarter if they are less predictable. One way to create that effect is to add nondeterminism to behavior trees.

For parts (a) and (b) below, use the expression child.execute() to invoke the Behavior Tree interpreter on a child node.

For parts (a) through (d), assume that the return values of True and False map onto Succeed and Fail, respectively. Also, assume the decorator Until Fail returns true when its child node fails, and Until Succeed returns true when its child succeeds.

(a) **2 points.** Write pseudo-code that implements a *Random Selector*, which is identical to a normal selector except that it considers its children in a random order, vs a fixed order.

(b) **2 points.** Write pseudo-code for a *Random Sequence*, which is identical to a normal Sequence, except it considers its children in a random order, vs a fixed order.

(c) **6 points.** Use these new node types together any of the following (Sequence; Selector; actions; checks; and the decorators Inverter, Until Fail, Until Succeed, Persist) to draw a behavior tree with this specification:

- Your character needs to move through a room. That room might contain a single enemy.
- The character considers fighting the enemy and crossing the room without a fight in some random order.
- Creeping through the room is only relevant if the character hears but does not see the enemy. Moving through the room is always relevant. Creeping is preferred.
- Fighting is only relevant if the enemy is visible.
- To fight, the character will hit the enemy, pause, and check to see if the enemy is conscious in a random order. This continues until the enemy is unconscious.
- When the enemy is unconscious, the character ties it up.
- On completion of this behavior, the character has either (a) crossed the room, or (b) is in the room facing a restrained and unconscious enemy.

In your drawing, denote a Random Selector by

and a Random Sequence by

Insert an extra sheet with your answer.

(d) **10 points.** We built a HFSM for a trash collecting robot in class. This question asks you to define a portion of its activity as a behavior tree with parallel branches.

We define a parallel node as a non-primitive node with an arbitrary number of children. All of those children will be tried once, and all must succeed for the parallel node to succeed. All of the children are launched in parallel. The moment any child returns Fail, the parallel node returns Fail.

Draw a behavior tree with the following specification:

- The robot is either tidying up trash, or recharging.
- Tidying is only relevant so long as trash is visible.
- Recharging is only relevant so long as trash is not visible.
- The tidying action never terminates of its own (it neither returns Success or Failure).
- The recharging action never terminates on its own (it neither returns Success or Failure).
- The robot continues to execute its tidying and recharging behavior forever.

Denote a parallel node by

As before, you may use Sequence; Selector; actions; checks; and the decorators Inverter, Until Fail, Until Succeed in your tree.

Draw your tree here or insert and extra sheet with your answer.

CM 146, Midterm
20 points
Note: Your responses for this problem should be attempted in Python for clarity. However, you will not be penalized for syntactic errors.

4. Consider the implementation of the Monte Carlo Tree Search below in order to solve this question for some imagined *single-player* game.

```
class Game:
     def __init__(self, *params): pass
     def legal_actions(self): pass
     def apply_move(self, move): pass
     def is_terminal(self): pass
     def score(self): pass

class Node:
     def __init__(self, last_action, action_list,
parent_node=None):
          self.child_nodes = {}
          self.score = 0
          self.visits = 0
          self.parent = parent_node
          self.parent_action = last_action
          self.untried_actions = action_list

def ucb(child_node):
     parent_node = child_node.parent
     return (child_node.score/child_node.visits) + \
          C * sqrt(log(parent_node.visits)/child_node.visits)
          # C is the exploration/exploitation constant

def traverse_nodes(node, game):
     while not game.is_terminal() and not node.untried_actions:
          child_node = max(node.child_nodes.values(), key = ucb)
          game.apply_move(child_node.parent_action)
          node = child_node
     return node

def backpropagate(score, node):
     while node:
          node.score += score
          node.visits += 1
          node = node.parent
```

(a) **6 points.** Many MCTS implementations exhaust untried actions before selecting a child node using the UCB algorithm. Partial expansion is an MCTS modification that lets child nodes compete with a parent node that still has untried actions. It is used primarily for games with large branching factors, i.e. having numerous possible legal actions at every state of the game. One approach to partial expansion, called *first play urgency* assigns a fixed value to untried actions and compares it to the results of the ucb calculation applied to previously visited (child) nodes.

Let *game.untriedActionValue* be the fixed value for scoring untried actions.

Rewrite traverse_nodes to implement first play urgency.

(b) **3 points.** Describe the impact of first play urgency on MCTS' search behavior as you dial up and down the value of untried actions.

(c) **8 points.** "Mixmax rewards" is a modification to MCTS that seeks to retain information regarding strong outcomes seen in nodes farther down the tree. It requires two changes:

- Each MCTS node must store the maximum of the win rates of its immediate children.
- The value associated with a node for selection - formerly its win rate - is now calculated as 80% of its win rate + 20% of the maximum win rate of its immediate children.

Change the appropriate functions to implement mixmax rewards. Assume search tree nodes have an additional field called "max_win_rate", which must be set.

(d) **3 points.** Describe the behavior of MCTS with mixmax rewards as you vary the relative importance of a node's win rate and the maximum win rate of its children.

Please insert extra sheet(s) with your answers to 4a – 4d.

CM 146, Midterm
20 points

5. Short Questions.
Insert extra sheet(s) with your answers.

(a) **2 points.** Given a 2D square grid that allows only 4 directions of movement (left, right, up, down), a starting cell (x1,y1), a goal cell (x2, y2), and a current cell (x,y). Which of these heuristics for path search makes A* explore the least number of tiles:

1. (x2 - x)^2 + (y2 - y)^2
2. abs(x2 - x) + abs(y2 - y)
3. abs(x2 - x)
4. abs(y2 - y)
5. Euclidean distance to the goal

(b) **2 points**. Reduced planning graphs improved the efficiency of Goal Oriented Action Planning techniques by 1-2 orders of magnitude. In 1-2 sentences explain why.

(c) **3 points.** Consider the following rules, which were present in PromWeek:

Friends is a reflexive relationship
(Friend ?x ?y) !(Friend ?y ?x) => (Friend ?y ?x)

Enemies is a reflexive relationship
(Enemy ?x ?y) !(Enemy ?y ?x) => (Enemy ?y ?x)

The Enemy of my friend is my enemy
(Friends ?x ?y) (Enemies ?y ?z) !(Enemies ?x ?z) => (Enemies ?x ?z)

Assume that working memory contains the following instances:

(Friend Spiderman PeterParker)
(Friend Goblin HarryOsborn)
(Enemies Spiderman Goblin)

What are the contents of working memory after running the rules in the order shown?

(d) **2 points**. One common heuristic in constraint satisfaction systems is to consider the most constrained variable next, for example, to select a cell to complete in Sudoku that has 2 vs 9 possible values. In 1-2 sentences, explain why this makes the search more efficient.

(e) **3 points.** You need to decide what classes to take next quarter. You must take 3 classes out of the following list.

Introduction to belly-button gazing (IBBG)	MWF	12:50 - 13:50

Advanced pumpkin carving (APC)	WTh	13:30 - 15:00
Introduction to the Introduction to Symmetry and its Applications to Quantum Mechanics (IIQ)	TTh	15:00 - 16:30
Binge-watching Scavengers Reign (SR)	WF	14:00 - 23:59
Star Trek: Theory and Applications (STTA)	MWF	14:00 - 15:00

Formulate your scheduling problem as a constraint satisfaction task. Show variables, domains, and describe at least two constraints. You can describe constraints in English.

Note: you do not have to solve the constraint satisfaction problem, just express it.

(f) **2 points.** Pyhop is a problem decomposition planner that searchers an AND/OR tree of problem decompositions. The infinite axis utility system ranks action options give features of the domain. What roles could an infinite axis utility model play within a problem decomposition planner like Pyhop? Select all that apply.

1. Guide the consideration order of subtasks within a method.
2. Guide the consideration order of methods for a task.
3. Choose which subtasks within a task to consider at all.
4. Choose which methods to consider at all.
5. None of the above: utility models are about action not mental search.

(g) **2 points**. Consider a genetic algorithm system that simultaneous evolves a population of Mario levels to defeat 50% of synthetic players, and a population of synthetic players to win 50% of the Mario levels. (The fitness functions penalize deviation from 50%). Let the system run until the fitness functions for players and levels don't significantly change across generations. On termination, which of the following might be true of the final populations? Check all that apply.

1. The average synthetic Mario player is excellent, and the average Mario level is hard by human standards.
2. The average synthetic Mario player is terrible, and the average Mario level is simple by human standards.
3. All synthetic players beat the most fit Mario level.
4. All Mario levels defeat the least fit synthetic player.

(h) **2 points.** Deep learning systems generally require a great deal of training data, but that data is frequently unavailable in many tasks. Which of the following are successful strategies for addressing this gap? Check all that apply.

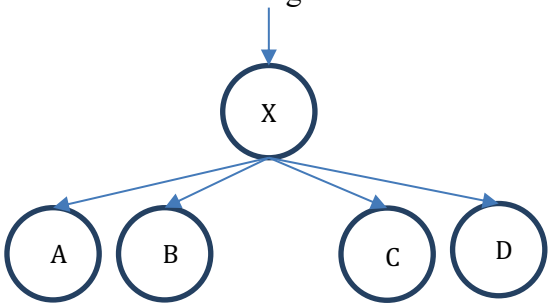
1. Incorporating neural network architecture and learned parameters from a related task.
2. Dialing up the learning rate.
3. Augmented the data by applying label-preserving transformations.

4. Reducing the size of the solution architecture.
5. Training longer (employing more training epochs) with the available data.

(i) **2 points.** Deep learning systems are subject to overfitting – the condition where further training *decreases* performance on the validation and test sets while *increasing* performance on the training set. What techniques reduce overfitting, allowing increased performance of the learned model against test data? Check all that apply.

1. Incorporating dropout layers.
2. Decreasing the size of the solution architecture.
3. Increasing the size of the solution architecture.
4. Ceasing training when overfitting occurs.
5. Modifying the loss function to smooth decision boundaries.
6. Increasing the learning rate.

Extra Credit
7 points
Consider the following behavior tree:



X is either a Sequence or Selector node (you do not know which). **A**, **B**, **C**, and **D**, are arbitrary subtrees.

Let **O** be an optional behavior, whose introduction would have no impact on the semantics of the above tree – its Success or Failure, the execution order of its components, or the number of times they are executed.

(a) **2 points.** An unnamed Professor claims that you cannot introduce a subtree between **B** and **C** that encodes an optional behavior, **O**, while leaving the remainder of the tree unchanged (and without altering the implementation of the node type **X**). Is this statement true, or false? Explain why in one or two sentences.

(b) **5 points.** Suppose you could transform the tree shown above into another tree by duplicating structure, rewiring structure, and/or introducing additional Sequence and Selector nodes. Draw a transformed tree that encodes an optional behavior, **O**, that will be considered after **B** and before **C**, while preserving the semantics of the original tree.