

Write answers in spaces provided.

Partial Credit: If you show your work and briefly describe your approach, we will happily give partial credit where possible. Answers without supporting work (or that are not clear/legible) may not be given credit. We also reserve the right to take off points for overly long answers.

Pseudocode: Pseudocode can be written at the level discussed in class and does not necessarily need to conform to any particular programming language or API.

Q1-2. Search: Algorithms	/4
Q3. Search: Heuristic Function Properties	/3
Q4-7. Search: Adversarial Search	/6
Q8. Games: Alpha-Beta Pruning	/3
Q9. MDPs: Mini-Gridworld	/4
Total	/20

Name: _____

1. (2 points) The following implementation of graph search may be incorrect. Circle all the problems with the code.

```

function GRAPH-SEARCH(problem, fringe)
  closed  $\leftarrow$  an empty set,
  fringe  $\leftarrow$  INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop
    if fringe is empty then
      return failure
    end if
    node  $\leftarrow$  REMOVE-FRONT(fringe)
    if GOAL-TEST(problem, STATE[node]) then
      return node
    end if
    ADD STATE[node] TO closed
    fringe  $\leftarrow$  INSERTALL(EXPAND(node, problem), fringe)
  end loop
end function

```

- (a) Nodes may be expanded twice.
 - (b) The algorithm is no longer complete.
 - (c) The algorithm could return an incorrect solution.
 - (d) None of the above.
2. (2 points) The following implementation of A* graph search may be incorrect. You may assume that the algorithm is being run with a consistent heuristic. Circle all the problems with the code.

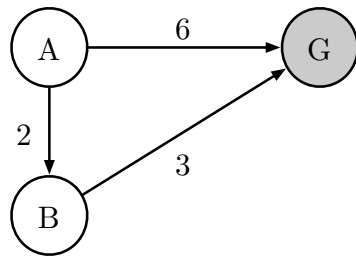
```

function A*-SEARCH(problem, fringe)
  closed  $\leftarrow$  an empty set
  fringe  $\leftarrow$  INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop
    if fringe is empty then
      return failure
    end if
    node  $\leftarrow$  REMOVE-FRONT(fringe)
    if STATE[node] IS NOT IN closed then
      ADD STATE[node] TO closed
      for successor IN GETSUCCESSORS(problem, STATE[node]) do
        fringe  $\leftarrow$  INSERT(MAKE-NODE(successor), fringe)
        if GOAL-TEST(problem, successor) then
          return successor
        end if
      end for
    end if
  end loop
end function

```

- (a) Nodes may be expanded twice.
- (b) The algorithm is no longer complete.
- (c) The algorithm does not always find the optimal solution.
- (d) None of the above.

3. For the following questions, consider the search problem shown on the left. It has only three states, and three directed edges. A is the start node and G is the goal node. To the right, four different heuristic functions are defined, numbered I through IV.



	$h(A)$	$h(B)$	$h(G)$
I	4	1	0
II	5	4	0
III	4	3	0
IV	5	2	0

- (a) **(2 points)** Admissibility and Consistency. For each heuristic function, circle whether it is admissible and whether it is consistent with respect to the search problem given above.

	Admissible?		Consistent?	
I	Yes	No	Yes	No
II	Yes	No	Yes	No
III	Yes	No	Yes	No
IV	Yes	No	Yes	No

- (b) **(1 points)** Function Domination. Recall that *domination* has a specific meaning when talking about heuristic functions.

Circle all true statements among the following.

- Heuristic function III dominates IV.
- Heuristic function IV dominates III.
- Heuristic functions III and IV have no dominance relationship.
- Heuristic function I dominates IV.
- Heuristic function IV dominates I.
- Heuristic functions I and IV have no dominance relationship.

4. (**2 points**) Write down pseudocode for Minimax Search. Make sure to include the base case.
Hint: You can define 3 functions: MINVALUE(s), MAXVALUE(s), VALUE(s)

5. (**2 points**) Write down pseudocode for Expectimax Search. Make sure to include the base case.
Hint: You can define 3 functions: $\text{EXPVALUE}(s)$, $\text{MAXVALUE}(s)$, $\text{VALUE}(s)$

6. (**1 points**) What is the $O(\quad)$ number of states expanded by Expectimax search, assuming there are b possible actions and the game ends after m moves (here you can assume $m = \text{number of agent moves} + \text{number of opponent moves}$)?

7. (**1 points**) Is exhaustive search with Expectimax feasible for Pacman? If not, how did we deal with this in Project #2?

8. **(3 points)** Assume we run $\alpha - \beta$ pruning expanding successors from left to right on a game with tree as shown in Figure 1 (a). Then we have that:

- (a) (*true or false*) For some choice of pay-off values, no pruning will be achieved (shown in Figure 1 (a)).
- (b) (*true or false*) For some choice of pay-off values, the pruning shown in Figure 1 (b) will be achieved.
- (c) (*true or false*) For some choice of pay-off values, the pruning shown in Figure 1 (c) will be achieved.
- (d) (*true or false*) For some choice of pay-off values, the pruning shown in Figure 1 (d) will be achieved.
- (e) (*true or false*) For some choice of pay-off values, the pruning shown in Figure 1 (e) will be achieved.
- (f) (*true or false*) For some choice of pay-off values, the pruning shown in Figure 1 (f) will be achieved.

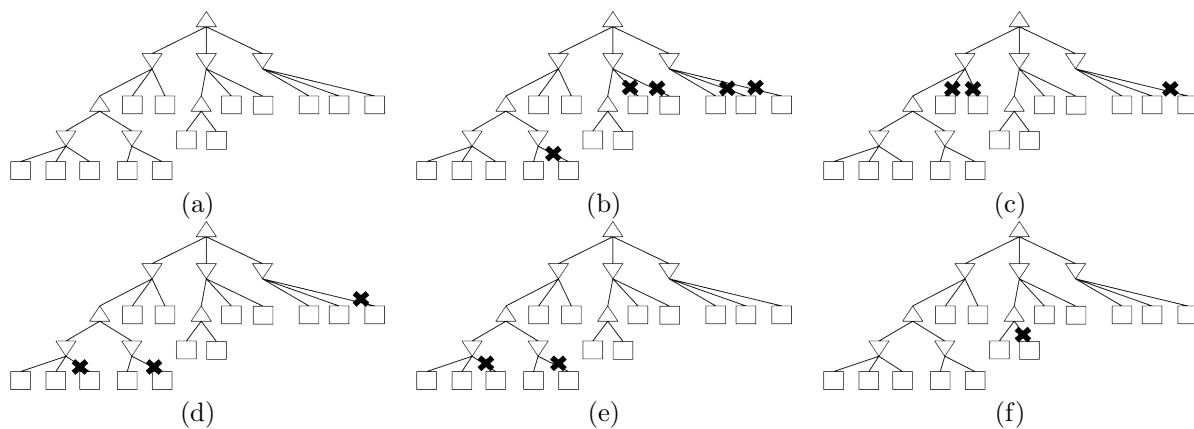
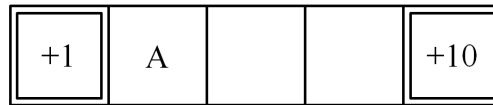


Figure 1: Game trees.

9. The following problems take place in various scenarios of the gridworld MDP. In all cases, A is the start state and double-rectangle states are exit states. From an exit state, the only action available is *Exit*, which results in the listed reward and ends the game (by moving into a terminal state X , not shown). From non-exit states, the agent can choose either *Left* or *Right* actions, which move the agent in the corresponding direction. There are no living rewards; the only non-zero rewards come from exiting the grid. Throughout this problem, assume that value iteration begins with initial values $V_0(s) = 0$ for all states s .

First, consider the following mini-grid. For now, the discount is $\gamma = 1$ and legal movement actions will always succeed (and so the state transition function is deterministic).



- (a) **(0.5 points)** What is the optimal value $V^*(A)$?
- (b) **(0.5 points)** When running value iteration, remember that we start with $V_0(s) = 0$ for all s . What is the first iteration k for which $V_k(A)$ will be non-zero?
- (c) **(0.5 points)** What will $V_k(A)$ be when it is first non-zero?
- (d) **(0.5 points)** After how many iterations k will we have $V_k(A) = V^*(A)$? If they will never become equal, write *never*.

Now the situation is as before, but the discount γ is less than 1.

- (e) **(1 points)** If $\gamma = 0.5$, what is the optimal value $V^*(A)$?
- (f) **(1 points)** For what range of values γ of the discount will it be optimal to go *Right* from A ? Remember that $0 \leq \gamma \leq 1$. Write *all* or *none* if all or no legal values of γ have this property.