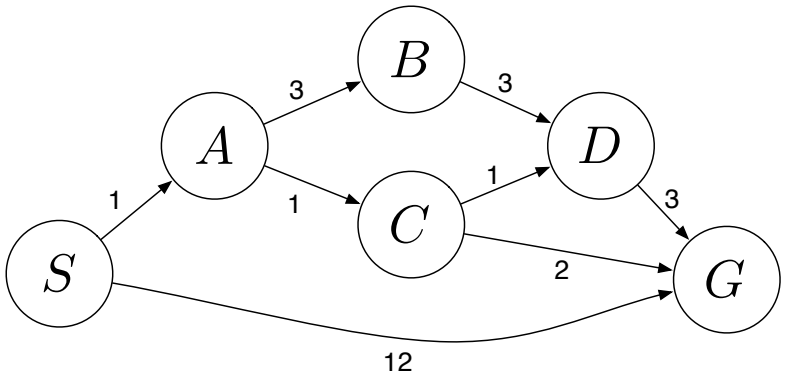
CS188 Fall 2018 Section 6: Midterm 1 Prep

1 . Search



Answer the following questions about the search problem shown above. Assume that ties are broken alphabetically. (For example, a partial plan $S \to X \to A$ would be expanded before $S \to X \to B$; similarly, $S \to A \to Z$ would be expanded before $S \to B \to A$.) For the questions that ask for a path, please give your answers in the form '$S - A - D - G$.'

(a) What path would breadth-first graph search return for this search problem?

(b) What path would uniform cost graph search return for this search problem?

(c) What path would depth-first graph search return for this search problem?

(d) What path would A* graph search, using a consistent heuristic, return for this search problem?

(e) Consider the heuristics for this problem shown in the table below.

State	h_1	h_2
S	5	4
A	3	2
B	6	6
C	2	1
D	3	3
G	0	0

(i) Is h_1 admissible? **Yes** **No**

(ii) Is h_1 consistent? **Yes** **No**

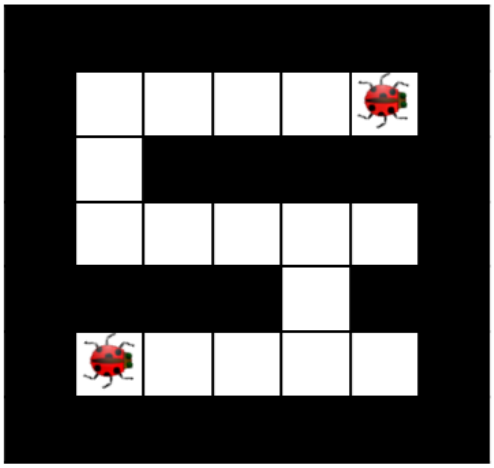
(iii) Is h_2 admissible? **Yes** **No**

(iv) Is h_2 consistent? **Yes** **No**

2 . Hive Minds: Redux

Let's revisit our bug friends. To recap, you control one or more insects in a rectangular maze-like environment with dimensions $M \times N$, as shown in the figures below. At each time step, an insect can move North, East, South, or West (but not diagonally) into an adjacent square if that square is currently free, or the insect may stay in its current location. Squares may be blocked by walls (as denoted by the black squares), but the map is known.

For the following questions, you should answer for a general instance of the problem, not simply for the example maps shown.



You now control a pair of long lost bug friends. You know the maze, but you do not have any information about which square each bug starts in. You want to help the bugs reunite. You must pose a search problem whose solution is an all-purpose sequence of actions such that, after executing those actions, both bugs will be on the same square, regardless of their initial positions. Any square will do, as the bugs have no goal in mind other than to see each other once again. Both bugs execute the actions mindlessly and do not know whether their moves succeed; if they use an action which would move them in a blocked direction, they will stay where they are. Unlike the flea in the previous question, bugs *cannot* jump onto walls. Both bugs can move in each time step. Every time step that passes has a cost of one.

(a) Give a *minimal* state representation for the above search problem.

(b) Give the size of the state space for this search problem.

(c) Give a nontrivial admissible heuristic for this search problem.

3 . CSPs: Time Management

Two of our TAs, Arjun and Dave, are making their schedules for a busy morning. There are five tasks to be carried out:

(F) Pick up food for the group's research seminar, which, sadly, takes one precious hour.
(H) Prepare homework questions, which takes 2 consecutive hours.
(P) Prepare the PR2 (robot that Pieter uses for research) for a group of preschoolers' visit, which takes one hour.
(S) Lead the research seminar, which takes one hour.
(T) Teach the preschoolers about the PR2 robot, which takes 2 consecutive hours.

The schedule consists of one-hour slots: 8am-9am, 9am-10am, 10am-11am, 11am-12pm. The requirements for the schedule are as follows:

1. In any given time slot each TA can do at most one task (F, H, P, S, T).
2. The PR2 preparation (P) should happen before teaching the preschoolers (T).
3. The food should be picked up (F) before the seminar (S).
4. The seminar (S) should be finished by 10am.
5. Arjun is going to deal with food pick up (F) since he has a car.
6. The TA not leading the seminar (S) should still attend, and hence cannot perform another task (F, T, P, H) during the seminar.
7. The seminar (S) leader does not teach the preschoolers (T).
8. The TA who teaches the preschoolers (T) must also prepare the PR2 robot (P).
9. Preparing homework questions (H) takes 2 consecutive hours, and hence should start at or before 10am.
10. Teaching the preschoolers (T) takes 2 consecutive hours, and hence should start at or before 10am.

To formalize this problem as a CSP, use the variables F, H, P, S and T. The values they take on indicate the TA responsible for it, and the starting time slot during which the task is carried out (for a task that spans 2 hours, the variable represents the starting time, but keep in mind that the TA will be occupied for the next hour also - make sure you enforce constraint (a)!). Hence there are eight possible values for each variable, which we will denote by A8, A9, A10, A11, D8, D9, D10, D11, where the letter corresponds to the TA and the number corresponds to the time slot. For example, assigning the value of A8 to a variables means that this task is carried about by Arjun from 8am to 9am.

(a) What is the size of the state space for this CSP?

(b) Which of the statements above include unary constraints?

(c) In the table below, enforce all unary constraints by crossing out values in the table on the left below. If you made a mistake, cross out the whole table and use the right one.

F	A8	A9	A10	A11	D8	D9	D10	D11
H	A8	A9	A10	A11	D8	D9	D10	D11
P	A8	A9	A10	A11	D8	D9	D10	D11
S	A8	A9	A10	A11	D8	D9	D10	D11
T	A8	A9	A10	A11	D8	D9	D10	D11

F	A8	A9	A10	A11	D8	D9	D10	D11
H	A8	A9	A10	A11	D8	D9	D10	D11
P	A8	A9	A10	A11	D8	D9	D10	D11
S	A8	A9	A10	A11	D8	D9	D10	D11
T	A8	A9	A10	A11	D8	D9	D10	D11

(d) Start from the table above, select the variable S and assign the value A9 to it. Perform forward checking by crossing out values in the table below. Again the table on the right is for you to use in case you believe you made a mistake.

F	A8	A9	A10	A11	D8	D9	D10	D11
H	A8	A9	A10	A11	D8	D9	D10	D11
P	A8	A9	A10	A11	D8	D9	D10	D11
S	A8	A9	A10	A11	D8	D9	D10	D11
T	A8	A9	A10	A11	D8	D9	D10	D11

F	A8	A9	A10	A11	D8	D9	D10	D11
H	A8	A9	A10	A11	D8	D9	D10	D11
P	A8	A9	A10	A11	D8	D9	D10	D11
S	A8	A9	A10	A11	D8	D9	D10	D11
T	A8	A9	A10	A11	D8	D9	D10	D11

(e) Based on the result of (d), what variable will we choose to assign next based on the MRV heuristic (breaking ties alphabetically)? Assign the first possible value to this variable, and perform forward checking by crossing out values in the table below. Again the table on the right is for you to use in case you believe you made a mistake.

Variable _____ is selected and gets assigned value _____.

F	A8	A9	A10	A11	D8	D9	D10	D11
H	A8	A9	A10	A11	D8	D9	D10	D11
P	A8	A9	A10	A11	D8	D9	D10	D11
S	A8	A9	A10	A11	D8	D9	D10	D11
T	A8	A9	A10	A11	D8	D9	D10	D11

F	A8	A9	A10	A11	D8	D9	D10	D11
H	A8	A9	A10	A11	D8	D9	D10	D11
P	A8	A9	A10	A11	D8	D9	D10	D11
S	A8	A9	A10	A11	D8	D9	D10	D11
T	A8	A9	A10	A11	D8	D9	D10	D11

Have we arrived at a dead end (i.e., has any of the domains become empty)?

(f) We return to the result from enforcing just the unary constraints, which we did in (c). Select the variable S and assign the value A9. Enforce arc consistency by crossing out values in the table below.

F	A8	A9	A10	A11	D8	D9	D10	D11
H	A8	A9	A10	A11	D8	D9	D10	D11
P	A8	A9	A10	A11	D8	D9	D10	D11
S	A8	A9	A10	A11	D8	D9	D10	D11
T	A8	A9	A10	A11	D8	D9	D10	D11

F	A8	A9	A10	A11	D8	D9	D10	D11
H	A8	A9	A10	A11	D8	D9	D10	D11
P	A8	A9	A10	A11	D8	D9	D10	D11
S	A8	A9	A10	A11	D8	D9	D10	D11
T	A8	A9	A10	A11	D8	D9	D10	D11

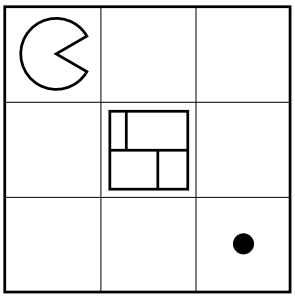
(g) Compare your answers to (d) and to (f). Does arc consistency remove more values or less values than forward checking does? Explain why.

(h) Check your answer to (f). Without backtracking, does any solution exist along this path? Provide the solution(s) or state that there is none.

4 . Surrealist Pacman

In the game of Surrealist Pacman, Pacman plays against a moving wall. On Pacman's turn, Pacman must move in one of the four cardinal directions, and must move into an unoccupied square. On the wall's turn, the wall must move in one of the four cardinal directions, and must move into an unoccupied square. The wall cannot move into a dot-containing square. Staying still is not allowed by either player. Pacman's score is always equal to the number of dots he has eaten.

The first game begins in the configuration shown below. Pacman moves first.



(a) Draw a game tree with one move for each player. Nodes in the tree represent game states (location of all agents and walls). Edges in the tree connect successor states to their parent states. Draw only the legal moves.

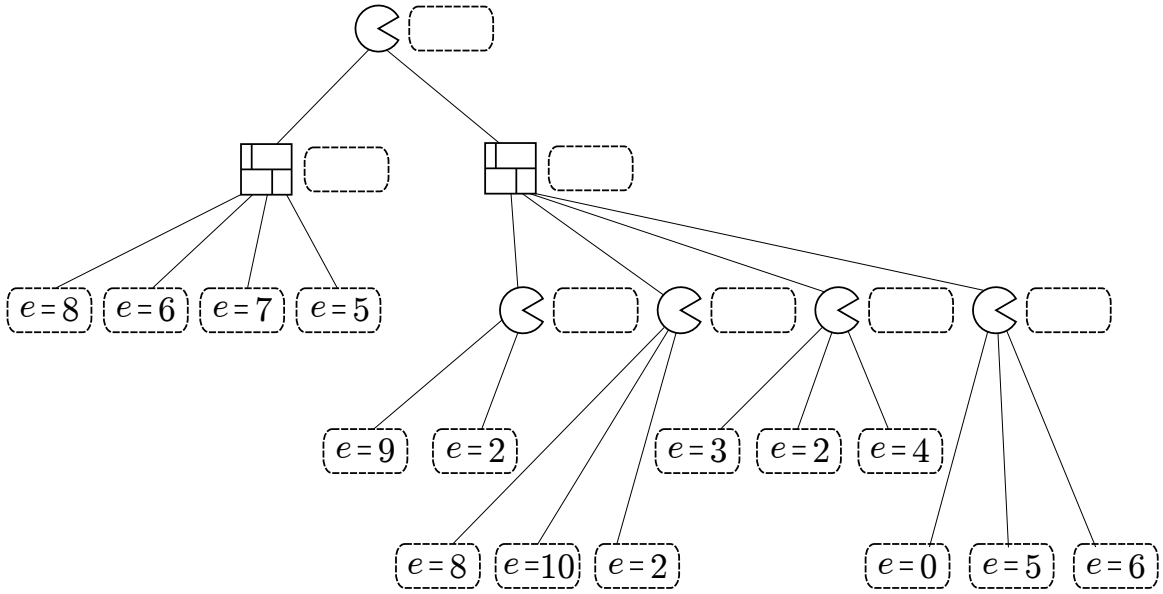
(b) According to the depth-limited game tree you drew above what is the value of the game? Use Pacman's score as your evaluation function.

(c) If we were to consider a game tree with ten moves for each player (rather than just one), what would be the value of the game as computed by minimax?

A second game is played on a more complicated board. A partial game tree is drawn, and leaf nodes have been scored using an (unknown) evaluation function e.

(d) In the dashed boxes, fill in the values of all internal nodes using the minimax algorithm.

(e) Cross off any nodes that are not evaluated when using alpha-beta pruning (assuming the standard left-to-right traversal of the tree).



Suppose that this evaluation function has a special property: it is known to give the correct minimax value of any internal node to within 2, and the correct minimax values of the leaf nodes exactly. That is, if v is the true minimax value of a particular node, and e is the value of the evaluation function applied to that node, $e-2 \leq v \leq e+2$, and $v = e$ if the node is a dashed box in the tree below.

Using this special property, you can modify the alpha-beta pruning algorithm to prune more nodes.

(f) Standard alpha-beta pseudocode is given below (only the max-value recursion). Fill in the boxes on the right to replace the corresponding boxes on the left so that the pseudocode prunes as many nodes as possible, taking account of this special property of the evaluation function.

```
function MAX-VALUE(node, α, β)
    e ← EVALUATIONFUNCTION(node)
    if node is leaf then
        return e
    end if
    [          ] (1)
    v ← −∞
    for child ← CHILDREN(node) do
        [v ← MAX(v, MIN-VALUE(child, α, β))] (2)
        if v ≥ β then
            return v
        end if
        α ← MAX(α, v)
    end for
    return v
end function
```

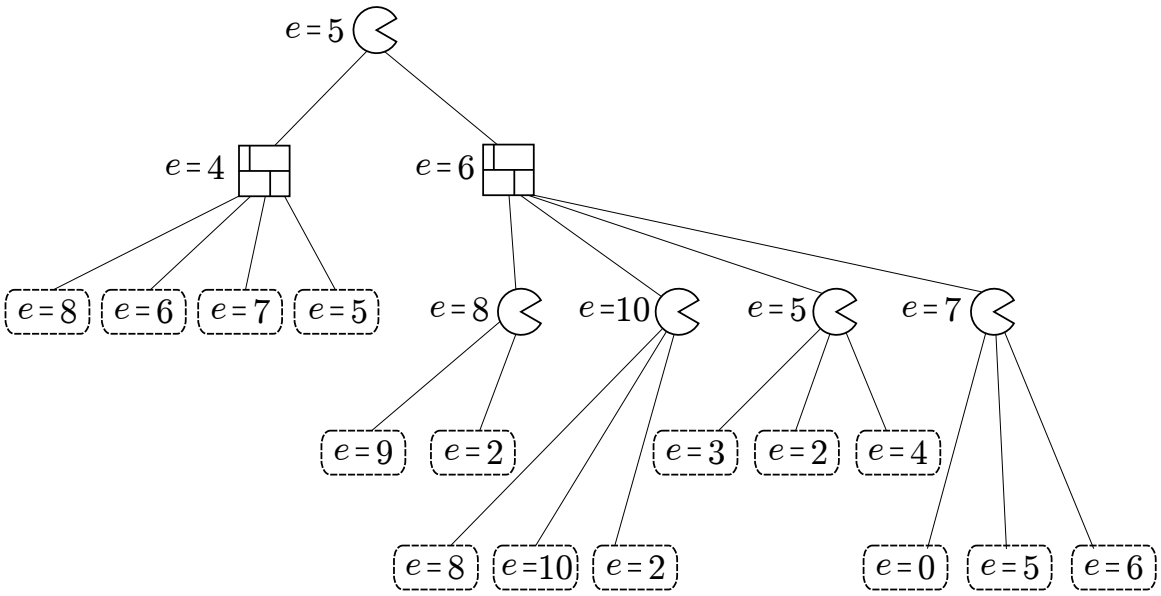
Fill in these boxes:

(1)

(2)

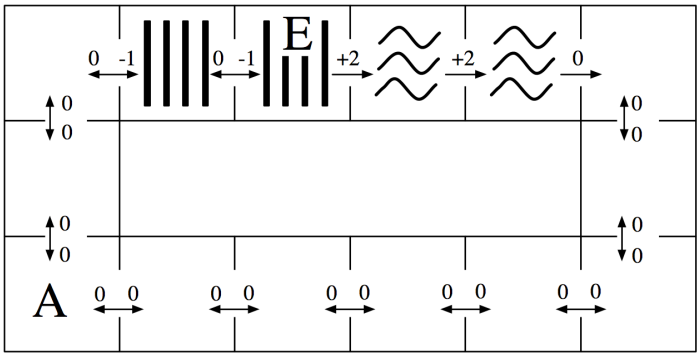
The same game tree is shown below, with the evaluation function applied to *internal* as well as leaf nodes.

(g) In the game tree below cross off any nodes that can be pruned assuming the special property holds true. If not sure you correctly formalized into pseudo-code your intuition on how to exploit the special property for improved pruning, make sure to annotate your pruned nodes with a brief explanation of why each of them was pruned.



5 . MDPs: Grid-World Water Park

Consider the MDP drawn below. The state space consists of all squares in a grid-world water park. There is a single waterslide that is composed of two ladder squares and two slide squares (marked with vertical bars and squiggly lines respectively). An agent in this water park can move from any square to any neighboring square, unless the current square is a slide in which case it must move forward one square along the slide. The actions are denoted by arrows between squares on the map and all deterministically move the agent in the given direction. The agent cannot stand still: it must move on each time step. Rewards are also shown below: the agent feels great pleasure as it slides down the water slide (+2), a certain amount of discomfort as it climbs the rungs of the ladder (-1), and receives rewards of 0 otherwise. The time horizon is infinite; this MDP goes on forever.

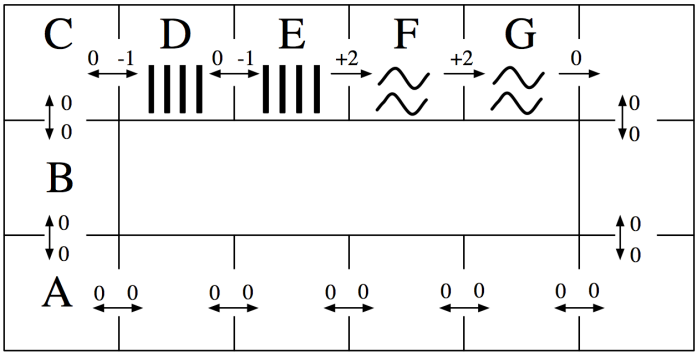


(a) How many (deterministic) policies π are possible for this MDP?

(b) Fill in the blank cells of this table with values that are correct for the corresponding function, discount, and state. *Hint: You should not need to do substantial calculation here.*

	γ	$s = A$	$s = E$
$V_3^*(s)$	1.0		
$V_{10}^*(s)$	1.0		
$V_{10}^*(s)$	0.1		
$Q_1^*(s, \text{west})$	1.0	——	
$Q_{10}^*(s, \text{west})$	1.0	——	
$V^*(s)$	1.0		
$V^*(s)$	0.1		

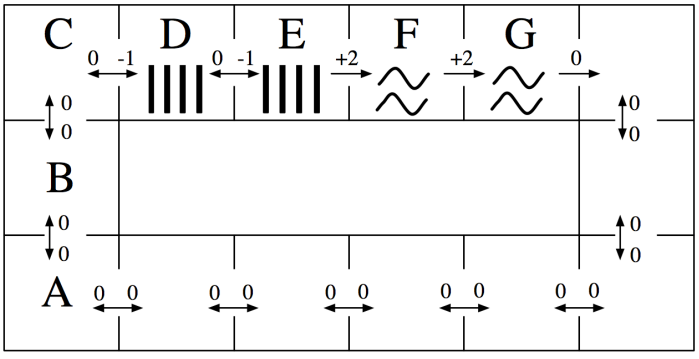
Use this labeling of the state space to complete the remaining subproblems:



(c) Fill in the blank cells of this table with the Q-values that result from applying the Q-update for the transition specified on each row. You may leave Q-values that are unaffected by the current update blank. Use discount $\gamma = 1.0$ and learning rate $\alpha = 0.5$. Assume all Q-values are initialized to 0. (Note: the specified transitions would not arise from a single episode.)

	$Q(D, \text{west})$	$Q(D, \text{east})$	$Q(E, \text{west})$	$Q(E, \text{east})$
Initial:	0	0	0	0
Transition 1: $(s = D, a = \text{east}, r = -1, s' = E)$				
Transition 2: $(s = E, a = \text{east}, r = +2, s' = F)$				
Transition 3: $(s = E, a = \text{west}, r = 0, s' = D)$				
Transition 4: $(s = D, a = \text{east}, r = -1, s' = E)$				

The agent is still at the water park MDP, but now we're going to use function approximation to represent Q-values. Recall that a policy π is *greedy* with respect to a set of Q-values as long as $\forall a, s\ Q(s, \pi(s)) \geq Q(s, a)$ (so ties may be broken in any way).



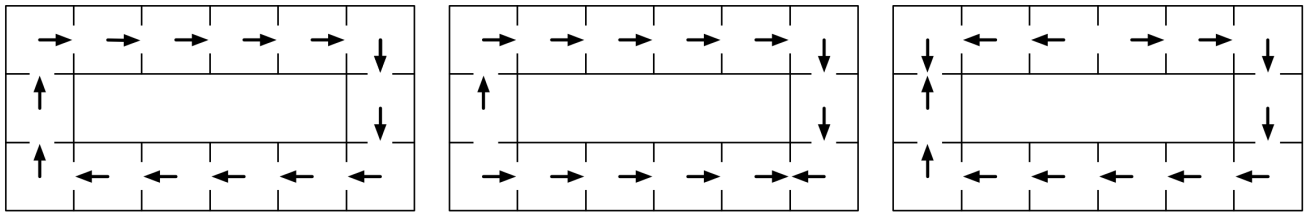
For the next subproblem, consider the following feature functions:

$$f(s,a) = \begin{cases} 1 & \text{if } a = \text{east}, \\ 0 & \text{otherwise.} \end{cases}$$

$$f'(s,a) = \begin{cases} 1 & \text{if } (a = \text{east}) \wedge \text{isSlide}(s), \\ 0 & \text{otherwise.} \end{cases}$$

(Note: isSlide(s) is true iff the state s is a slide square, i.e. either F or G.)

Also consider the following policies:



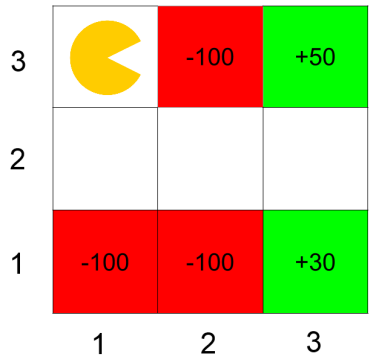
π_1 π_2 π_3

(d) Which are greedy policies with respect to the Q-value approximation function obtained by running the single Q-update for the transition $(s = F, a = \text{east}, r = +2, s' = G)$ while using the specified feature function? You may assume that all feature weights are zero before the update. Use discount $\gamma = 1.0$ and learning rate $\alpha = 1.0$. Circle all that apply.

f	π_1	π_2	π_3
f'	π_1	π_2	π_3

6 . Deep inside Q-learning

Consider the grid-world given below and an agent who is trying to learn the optimal policy. Rewards are only awarded for taking the *Exit* action from one of the shaded states. Taking this action moves the agent to the Done state, and the MDP terminates. Assume $\gamma = 1$ and $\alpha = 0.5$ for all calculations. All equations need to explicitly mention γ and α if necessary.



(a) The agent starts from the top left corner and you are given the following episodes from runs of the agent through this grid-world. Each line in an Episode is a tuple containing (s, a, s', r).

Episode 1	Episode 2	Episode 3	Episode 4	Episode 5
(1,3), S, (1,2), 0	(1,3), S, (1,2), 0	(1,3), S, (1,2), 0	(1,3), S, (1,2), 0	(1,3), S, (1,2), 0
(1,2), E, (2,2), 0	(1,2), E, (2,2), 0	(1,2), E, (2,2), 0	(1,2), E, (2,2), 0	(1,2), E, (2,2), 0
(2,2), E, (3,2), 0	(2,2), S, (2,1), 0	(2,2), E, (3,2), 0	(2,2), E, (3,2), 0	(2,2), E, (3,2), 0
(3,2), N, (3,3), 0	(2,1), Exit, D, -100	(3,2), S, (3,1), 0	(3,2), N, (3,3), 0	(3,2), S, (3,1), 0
(3,3), Exit, D, +50		(3,1), Exit, D, +30	(3,3), Exit, D, +50	(3,1), Exit, D, +30

Fill in the following Q-values obtained from direct evaluation from the samples:

$Q((3,2), \text{N}) =$ _____ $Q((3,2), \text{S}) =$ _____ $Q((2,2), \text{E}) =$ _____

(b) Q-learning is an online algorithm to learn optimal Q-values in an MDP with unknown rewards and transition function. The update equation is:

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a'))$$

where γ is the discount factor, α is the learning rate and the sequence of observations are $(\cdots, s_t, a_t, s_{t+1}, r_t, \cdots)$. Given the episodes in (a), fill in the time at which the following Q values first become non-zero. Your answer should be of the form (**episode#,iter#**) where **iter#** is the Q-learning update iteration in that episode. If the specified Q value never becomes non-zero, write *never*.

$Q((1,2), \text{E}) =$ _____ $Q((2,2), \text{E}) =$ _____ $Q((3,2), \text{S}) =$ _____

(c) In Q-learning, we look at a window of (s_t, a_t, s_{t+1}, r_t) to update our Q-values. One can think of using an update rule that uses a larger window to update these values. Give an update rule for $Q(s_t, a_t)$ given the window $(s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2})$.

$Q(s_t, a_t) =$

$Q(s_t, a_t) =$

$Q(s_t, a_t) =$