11.

Aim:

The aim of this code is to implement a Reinforcement Learning algorithm, specifically Q-learning, to teach an agent to navigate a gridworld and reach the winning state with the highest reward while avoiding the losing state.

Algorithm:

```
The algorithm used in this code is Q-learning. It is a model-free, off-policy algorithm that learns the optimal action-value function by iteratively updating the Q-values for each state-action pair using the Bellman equation: Q(s,a) = Q(s,a) + \text{alpha} * (\text{reward} + \text{gamma} * \text{max}(Q(s',a')) - Q(s,a)) Where:
```

```
Q(s,a) is the expected reward for taking action a in state s alpha is the learning rate reward is the immediate reward for taking action a in state s gamma is the discount factor max(Q(s',a')) is the maximum expected reward for taking any action in the next state s'
```

program

```
import numpy as np
# global variables
BOARD_ROWS = 3
BOARD_COLS = 4
WIN_STATE = (0, 3)
LOSE_STATE = (1, 3)
START = (2, 0)
DETERMINISTIC = True
class State:
    def __init__(self, state=START):
        self.board = np.zeros([BOARD_ROWS, BOARD_COLS])
        self.board[1, 1] = -1
```

```
self.state = state
  self.isEnd = False
  self.determine = DETERMINISTIC
def giveReward(self):
  if self.state == WIN_STATE:
    return 1
  elif self.state == LOSE_STATE:
    return -1
  else:
    return 0
def isEndFunc(self):
  if (self.state == WIN_STATE) or (self.state == LOSE_STATE):
    self.isEnd = True
def nxtPosition(self, action):
  if self.determine:
    if action == "up":
       nxtState = (self.state[0] - 1, self.state[1])
    elif action == "down":
       nxtState = (self.state[0] + 1, self.state[1])
    elif action == "left":
       nxtState = (self.state[0], self.state[1] - 1)
    else:
       nxtState = (self.state[0], self.state[1] + 1)
    if (nxtState[0] >= 0) and (nxtState[0] <= (BOARD_ROWS -1)):</pre>
       if (nxtState[1] >= 0) and (nxtState[1] <= (BOARD_COLS -1)):</pre>
         if nxtState != (1, 1):
           return nxtState
    return self.state
def showBoard(self):
  self.board[self.state] = 1
  for i in range(0, BOARD_ROWS):
    print('----')
    out = '| '
```

```
for j in range(0, BOARD_COLS):
         if self.board[i, j] == 1:
           token = '*'
         if self.board[i, j] == -1:
           token = 'z'
         if self.board[i, j] == 0:
           token = '0'
         out += token + ' | '
       print(out)
    print('----')
# Agent of player
class Agent:
  def __init__(self):
    self.states = []
    self.actions = ["up", "down", "left", "right"]
    self.State = State()
    self.lr = 0.2
    self.exp_rate = 0.3
    self.state_values = {}
    for i in range(BOARD_ROWS):
       for j in range(BOARD_COLS):
         self.state_values[(i, j)] = 0 # set initial value to 0
  def chooseAction(self):
    mx_nxt_reward = 0
    action = ""
    if np.random.uniform(0, 1) <= self.exp_rate:</pre>
       action = np.random.choice(self.actions)
    else:
       for a in self.actions:
         nxt_reward = self.state_values[self.State.nxtPosition(a)]
         if nxt_reward >= mx_nxt_reward:
           action = a
```

```
mx_nxt_reward = nxt_reward
  return action
def takeAction(self, action):
  position = self.State.nxtPosition(action)
  return State(state=position)
def reset(self):
  self.states = []
  self.State = State()
def play(self, rounds=10):
  i = 0
  while i < rounds:
    if self.State.isEnd:
      reward = self.State.giveReward()
      self.state_values[self.State.state] = reward
      print("Game End Reward", reward)
      for s in reversed(self.states):
        reward = self.state_values[s] + self.lr * (reward -self.state_values[s])
        self.state_values[s] = round(reward, 3)
      self.reset()
      i += 1
    else:
      action = self.chooseAction()
      self.states.append(self.State.nxtPosition(action))
      print("current position {} action {}".format(self.State.state,action))
      self.State = self.takeAction(action)
      self.State.isEndFunc()
      print("nxt state", self.State.state)
      print("----")
def showValues(self):
  for i in range(0, BOARD_ROWS):
    print('----')
    out = '| '
    for j in range(0, BOARD_COLS):
```

```
out += str(self.state_values[(i, j)]).ljust(6) + ' | '
    print(out)
    print('-----')

if __name__ == "__main__":
    ag = Agent()
    ag.play(50)
    print(ag.showValues())
```

Output:

```
current position (2, 0) action right
nxt state (2, 1)
current position (2, 1) action right
nxt state (2, 2)
current position (2, 2) action right
nxt state (2, 3)
current position (2, 3) action right
nxt state (2, 3)
current position (2, 3) action left
nxt state (2, 2)
current position (2, 2) action right
nxt state (2, 3)
current position (2, 3) action right
nxt state (2, 3)
```

current position (2, 3) acti	on right		
nxt state (2, 3)			
current position (2, 3) acti	on down		
nxt state (2, 3)			
current position (2, 3) acti	ion right		
nxt state (2, 3)			
current position (2, 3) acti	on right		
nxt state (2, 3)			
current position (2, 3) acti	ion right		
nxt state (2, 3)	<u></u>		
current position (2, 3) acti	on right		
nxt state (2, 3)			
current position (2, 3) acti	on left		
nxt state (2, 2)			
current position (2, 2) acti	ion right		
nxt state (2, 3)			
current position (2, 3) acti	ion left		
nxt state (2, 2)			
current position (2, 2) acti	on right		
nxt state (2, 3)			
current position (2, 2) acti	ion right		
current position (2, 3) acti	OITTIGITE		
nxt state (2, 3)			

current position (2, 3) action down
nxt state (2, 3)
current position (2, 3) action right
nxt state (2, 3)

current position (2, 3) action right
nxt state (2, 3)
current position (2, 3) action right
nxt state (2, 3)
aurrent nesition (2, 2) estion right
current position (2, 3) action right
nxt state (2, 3)
current position (2, 3) action down
nxt state (2, 3)

current position (2, 3) action right
nxt state (2, 3)

current position (2, 3) action right
nxt state (2, 3)

current position (2, 3) action right
nxt state (2, 3)
current position (2, 3) action right
nxt state (2, 3)
current position (2, 3) action right
nxt state (2, 3)

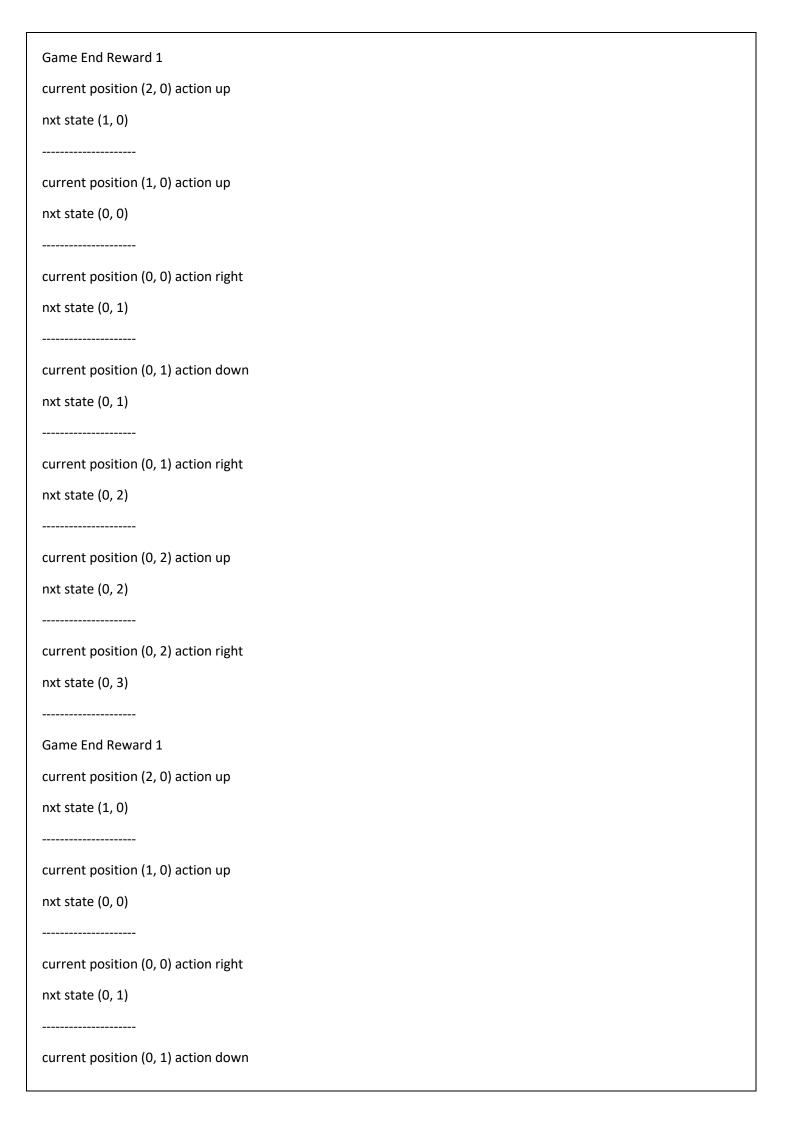
current position (2, 3) action right
nxt state (2, 3)
current position (2, 3) action right
nxt state (2, 3)

current position (2, 3) action right
nxt state (2, 3)
current position (2, 3) action right
nxt state (2, 3)
current position (2, 3) action right
nxt state (2, 3)

current position (2, 3) action up
nxt state (1, 3)
Game End Reward -1
current position (2, 0) action up
nxt state (1, 0)
current position (1, 0) action right
nxt state (1, 0)
current position (1, 0) action right
nxt state (1, 0)
current position (1, 0) action right
nxt state (1, 0)
current position (1, 0) action right
nxt state (1, 0)



nxt state (0, 0)		
current position (0, 0) action right		
nxt state (0, 1)		
current position (0, 1) action right		
nxt state (0, 2)		
current position (0, 2) action right		
nxt state (0, 3)		
Game End Reward 1		
current position (2, 0) action up		
nxt state (1, 0)		
current position (1, 0) action up		
nxt state (0, 0)		
current position (0, 0) action down		
nxt state (1, 0)		
current position (1, 0) action up		
nxt state (0, 0)		
current position (0, 0) action right		
nxt state (0, 1)		
current position (0, 1) action right		
nxt state (0, 2)		
current position (0, 2) action right		
nxt state (0, 3)		



nxt state (0, 1)		
current position (0, 1) action right		
nxt state (0, 2)		
current position (0, 2) action right		
nxt state (0, 3)		
Game End Reward 1		
current position (2, 0) action up		
nxt state (1, 0)		
current position (1, 0) action down		
nxt state (2, 0)		
current position (2, 0) action up		
nxt state (1, 0)		
current position (1, 0) action right		
nxt state (1, 0)		
current position (1, 0) action down		
nxt state (2, 0)		
current position (2, 0) action up		
nxt state (1, 0)		
current position (1, 0) action up		
nxt state (0, 0)		
current position (0, 0) action right		
nxt state (0, 1)		







nxt state (0, 1)	
current position (0, 1) action right	
nxt state (0, 2)	
current position (0, 2) action right	
nxt state (0, 3)	
Game End Reward 1	
current position (2, 0) action left	
nxt state (2, 0)	
current position (2, 0) action down	
nxt state (2, 0)	
current position (2, 0) action up	
nxt state (1, 0)	
current position (1, 0) action up	
nxt state (0, 0)	
current position (0, 0) action down	
nxt state (1, 0)	
current position (1, 0) action up	
nxt state (0, 0)	
current position (0, 0) action right	
nxt state (0, 1)	
current position (0, 1) action up	
nxt state (0, 1)	
	

current position (0, 1) action down
nxt state (0, 1)
current position (0, 1) action down
nxt state (0, 1)
current position (0, 1) action right
current position (0, 1) action right nxt state (0, 2)
current position (0, 2) action right
nxt state (0, 3)
Game End Reward 1
current position (2, 0) action left
nxt state (2, 0)

current position (2, 0) action up
nxt state (1, 0)
current position (1, 0) action left
nxt state (1, 0)

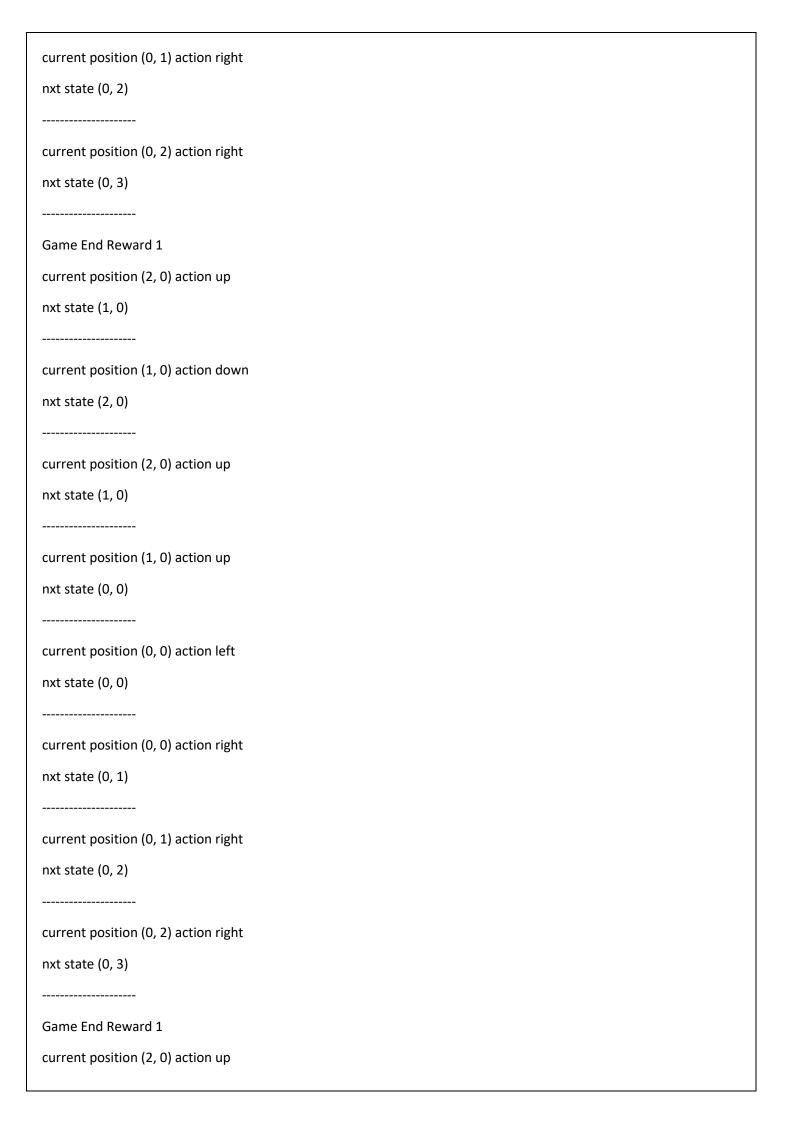
current position (1, 0) action up
nxt state (0, 0)
current position (0, 0) action right
nxt state (0, 1)

current position (0, 1) action right
nxt state (0, 2)
current position (0, 2) action up
current position (0, 2) action up nxt state (0, 2)
TIAL SLUCE (O, 2)

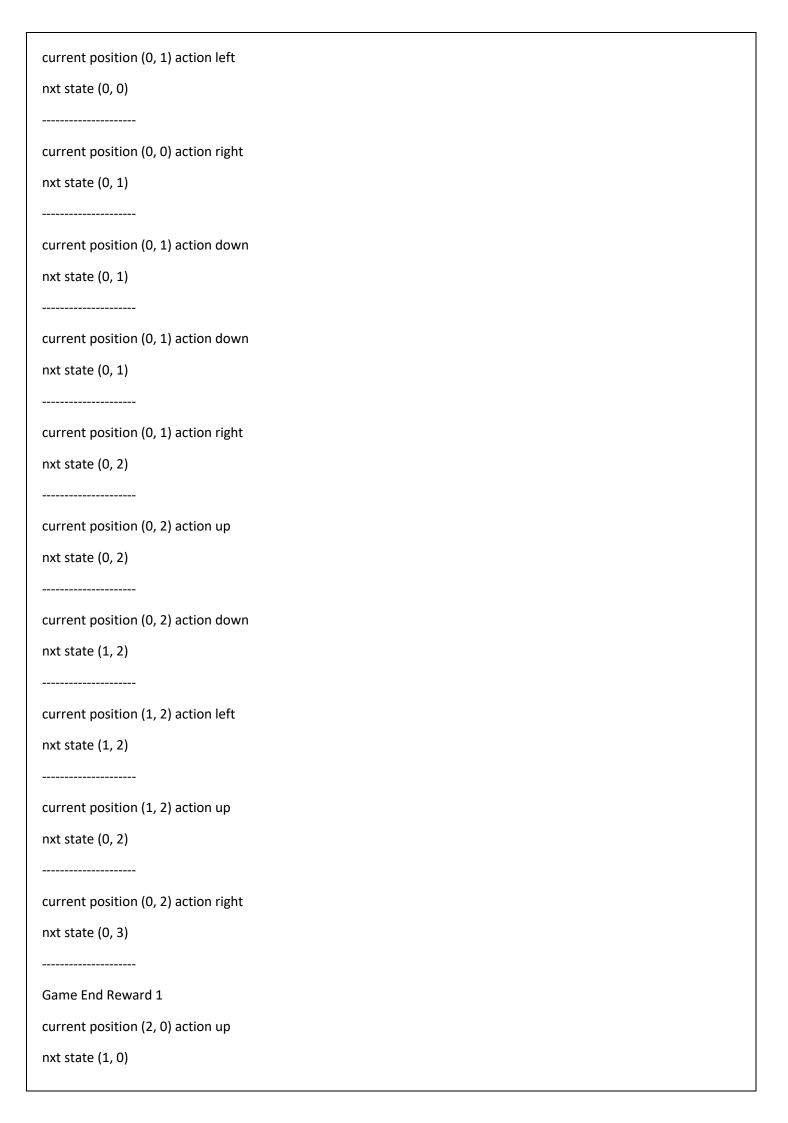




nxt state (1, 0)			
current position (1, 0) acti	ion up		
nxt state (0, 0)			
current position (0, 0) acti	ion right		
nxt state (0, 1)			
current position (0, 1) acti	ion right		
nxt state (0, 2)			
current position (0, 2) acti	ion right		
nxt state (0, 3)			
Game End Reward 1			
current position (2, 0) acti	ion down		
nxt state (2, 0)			
current position (2, 0) acti	ion left		
nxt state (2, 0)			
current position (2, 0) acti	ion up		
nxt state (1, 0)			
current position (1, 0) acti	ion up		
nxt state (0, 0)			
current position (0, 0) acti	ion up		
nxt state (0, 0)			
current position (0, 0) acti	ion right		
nxt state (0, 1)			







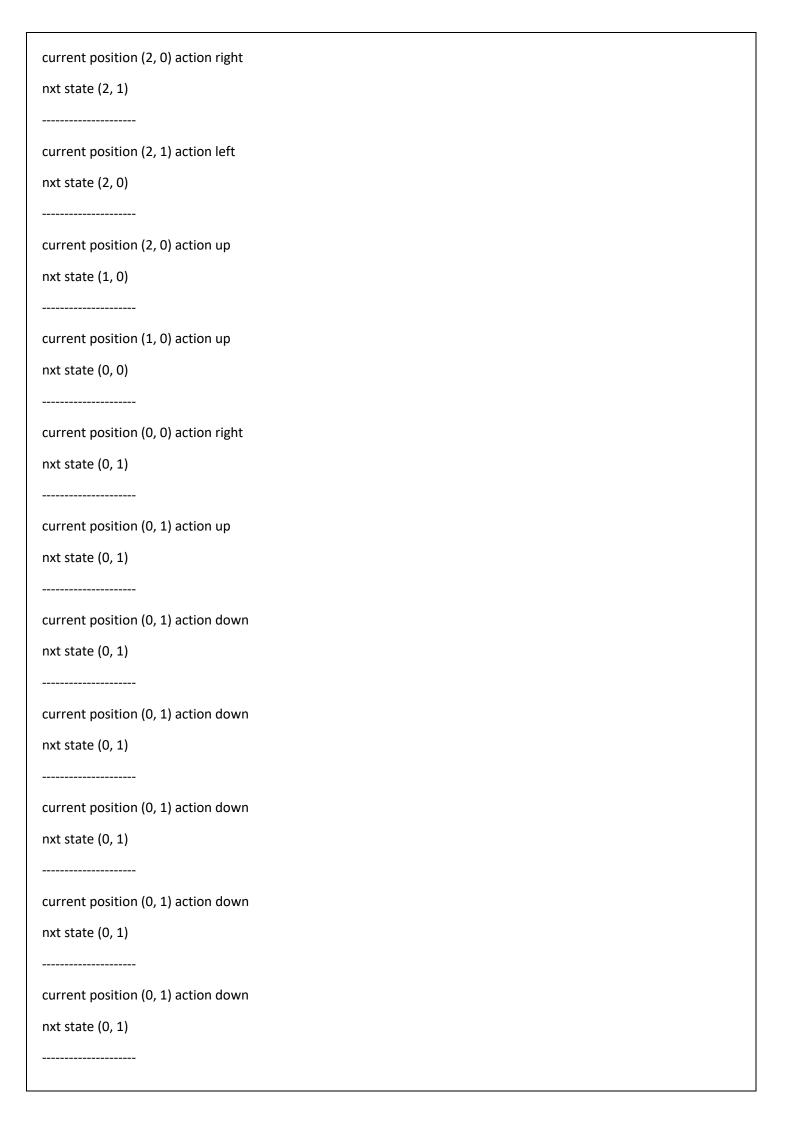










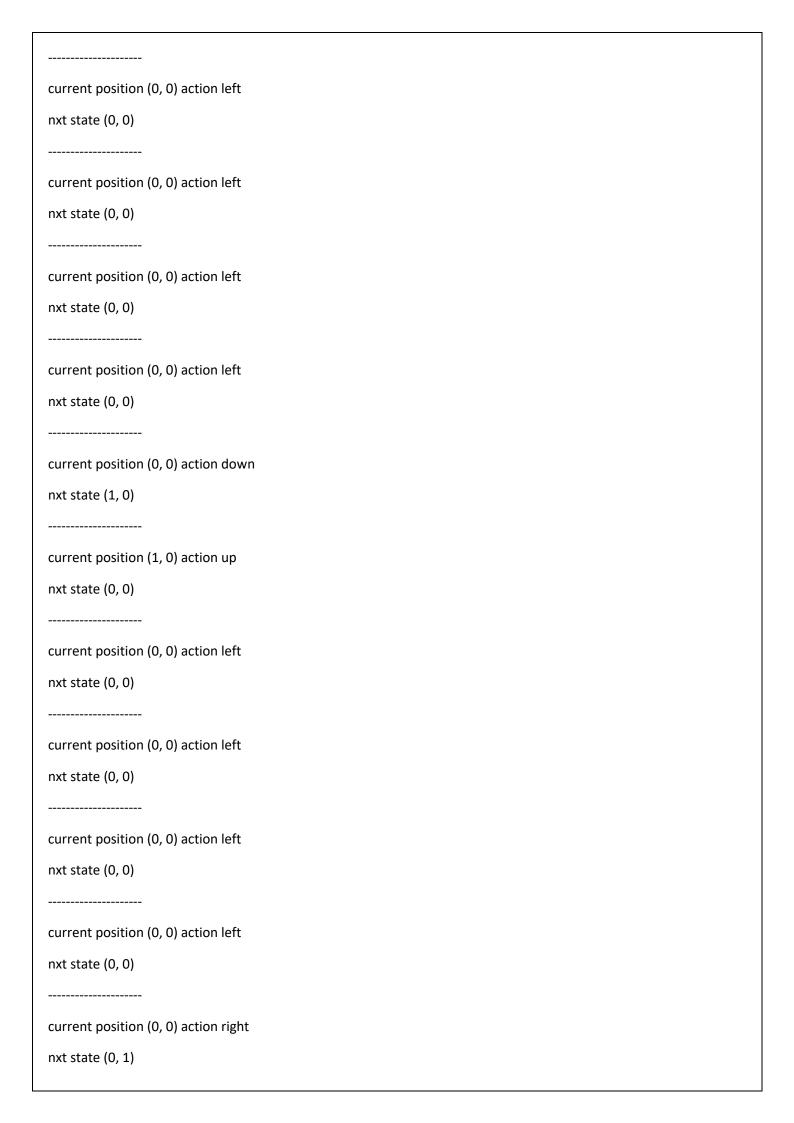


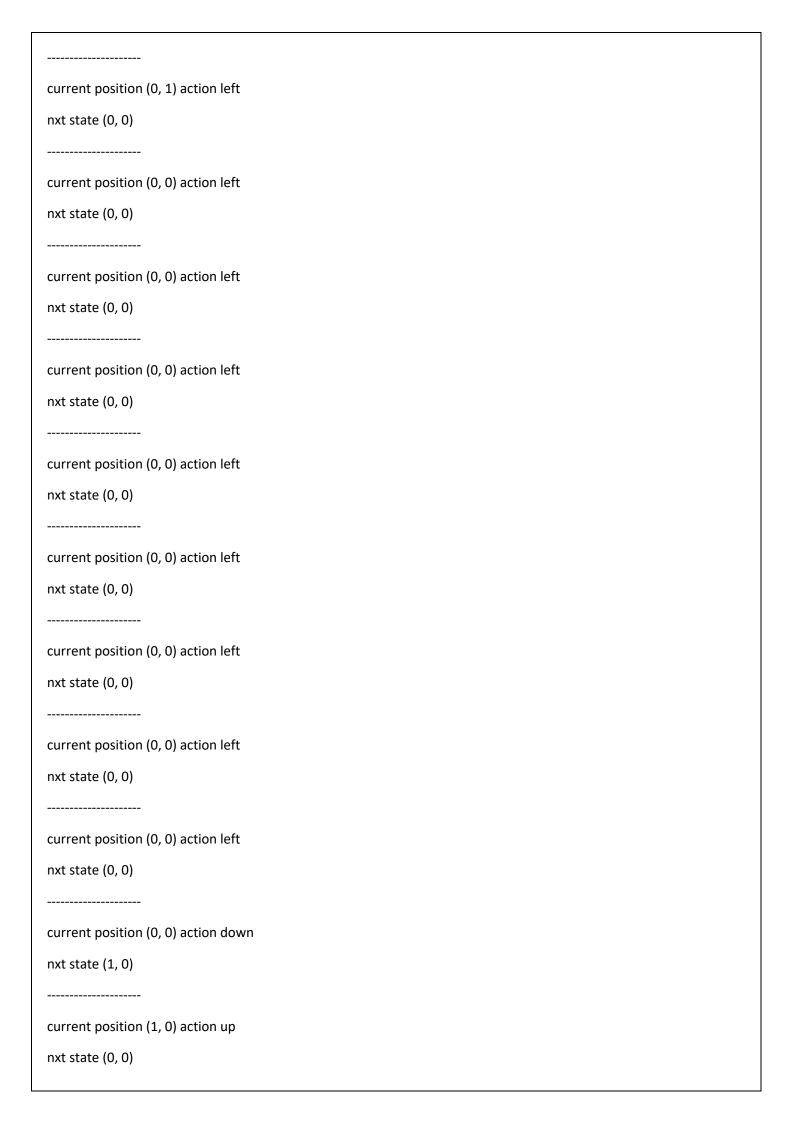
current position (0, 1) action down
nxt state (0, 1)
current position (0, 1) action down
nxt state (0, 1)
current position (0, 1) action right
nxt state (0, 2)
current position (0, 2) action up
nxt state (0, 2)
current position (0, 2) action right
nxt state (0, 3)
Game End Reward 1
current position (2, 0) action up
nxt state (1, 0)
current position (1, 0) action left
nxt state (1, 0)
current position (1, 0) action up
nxt state (0, 0)

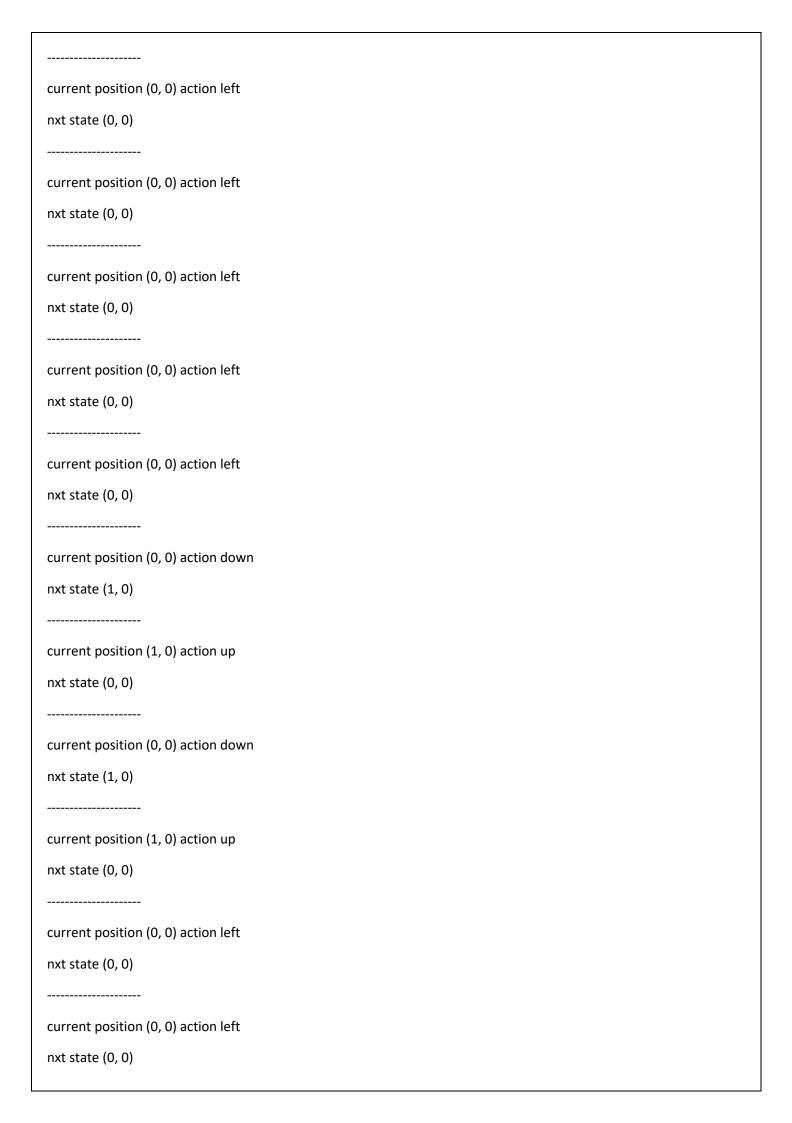
current position (0, 0) action left
nxt state (0, 0)

current position (0, 0) action left
nxt state (0, 0)

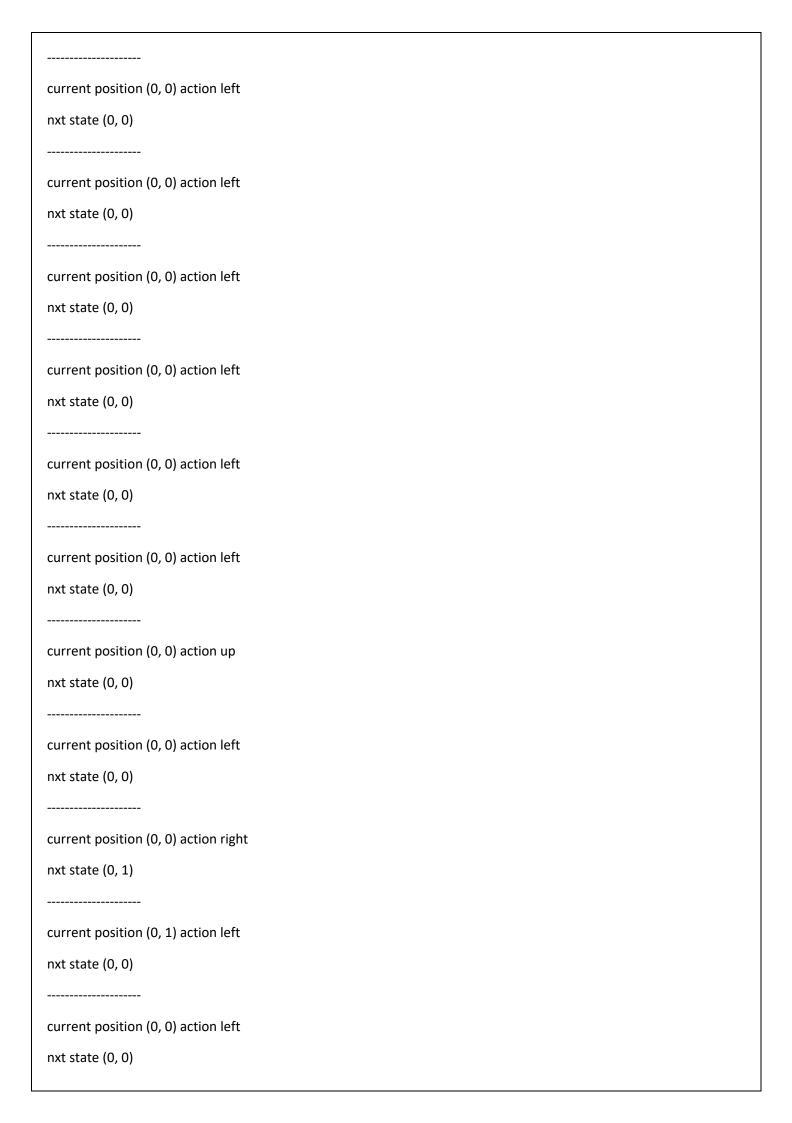
current position (0, 0) action left
nxt state (0, 0)

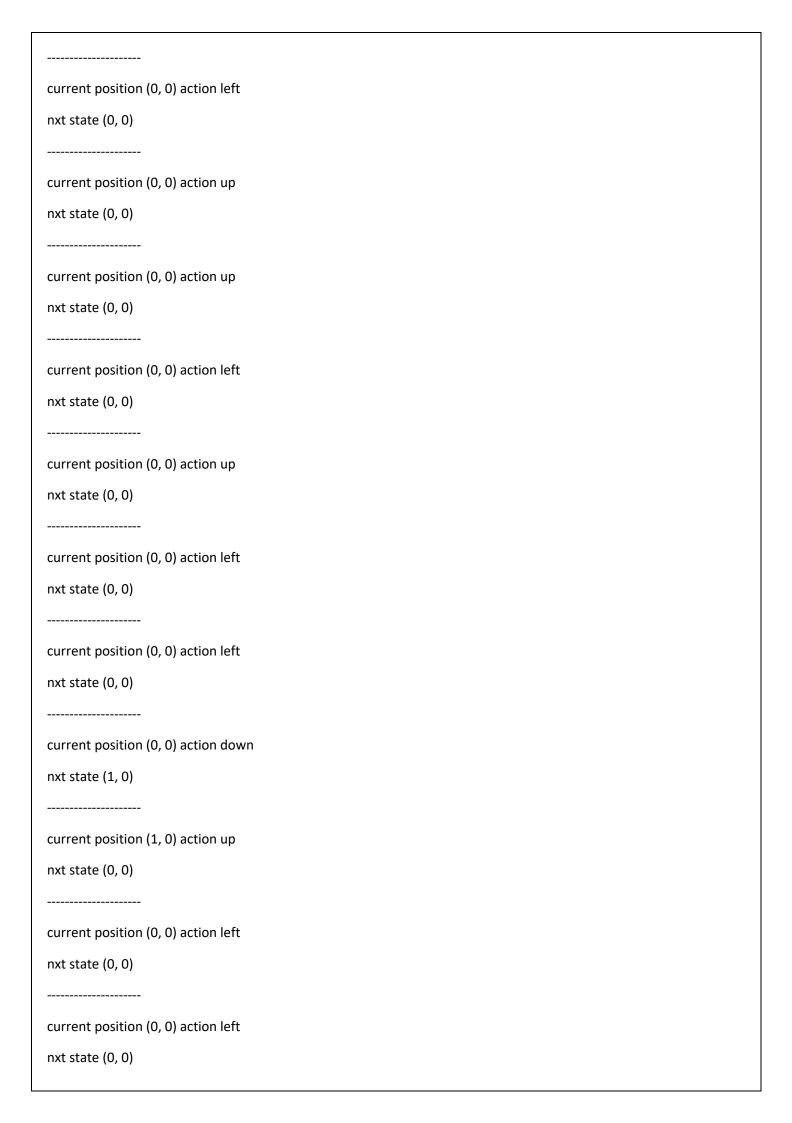


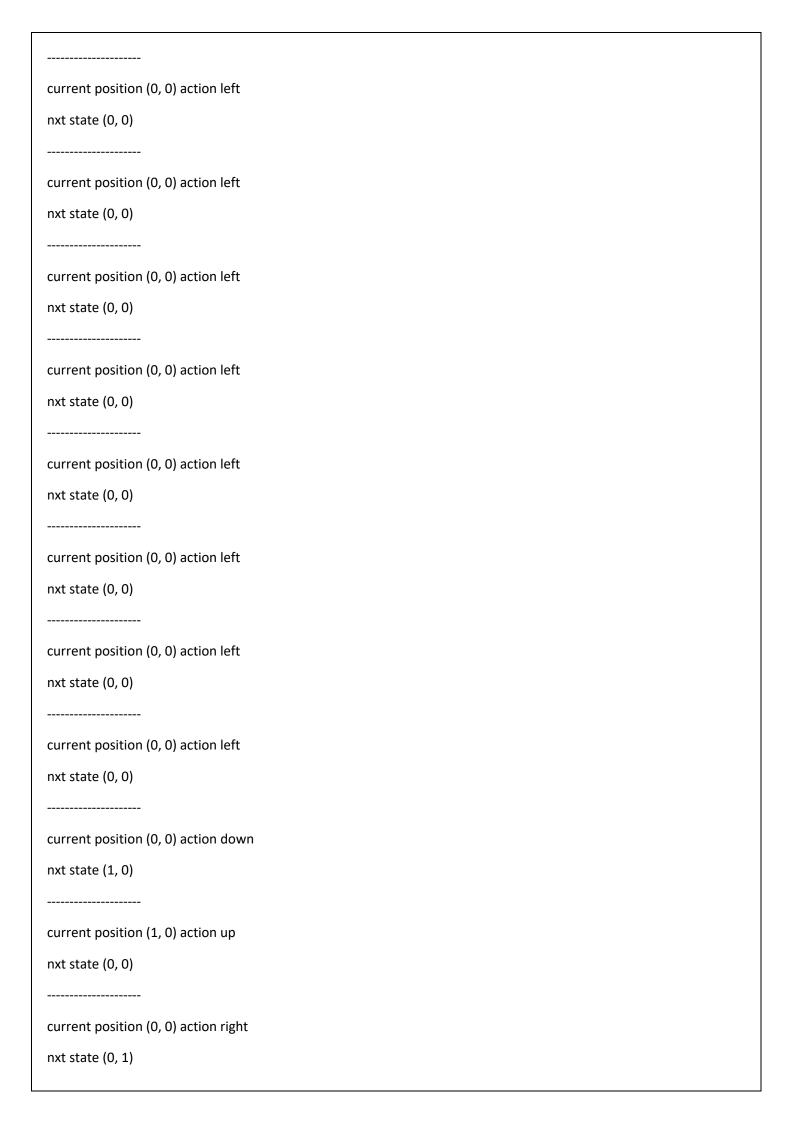


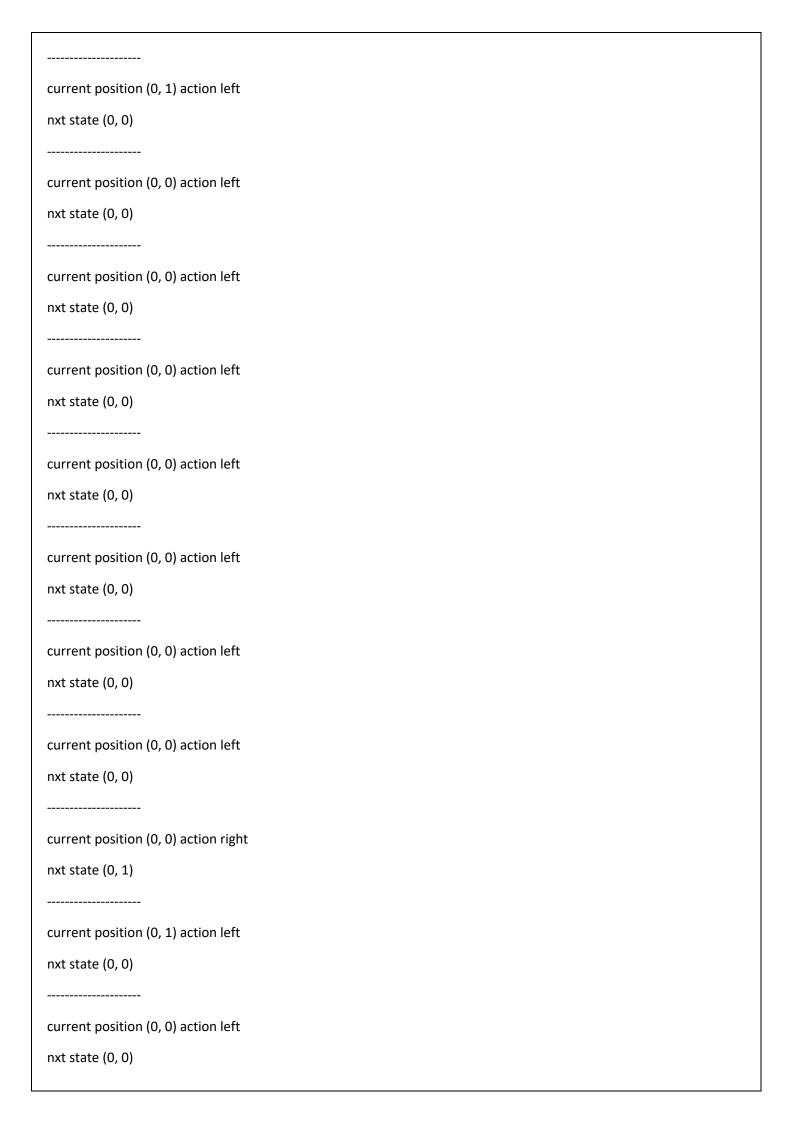


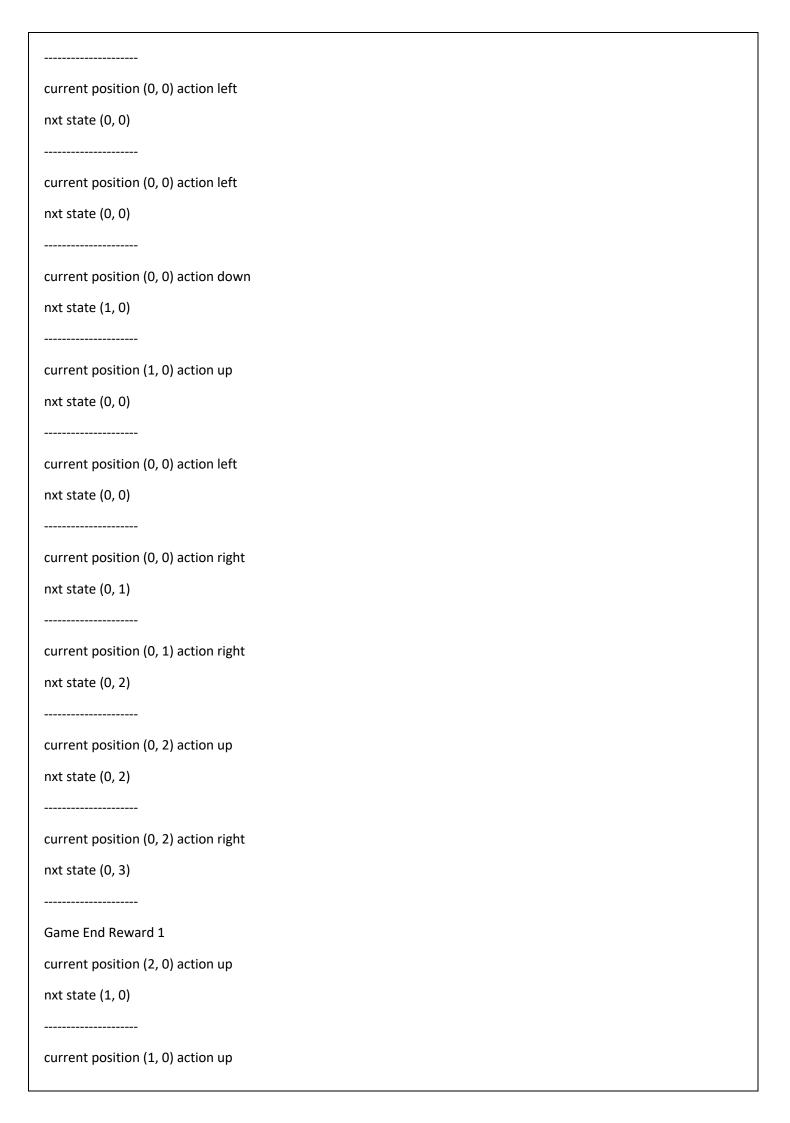




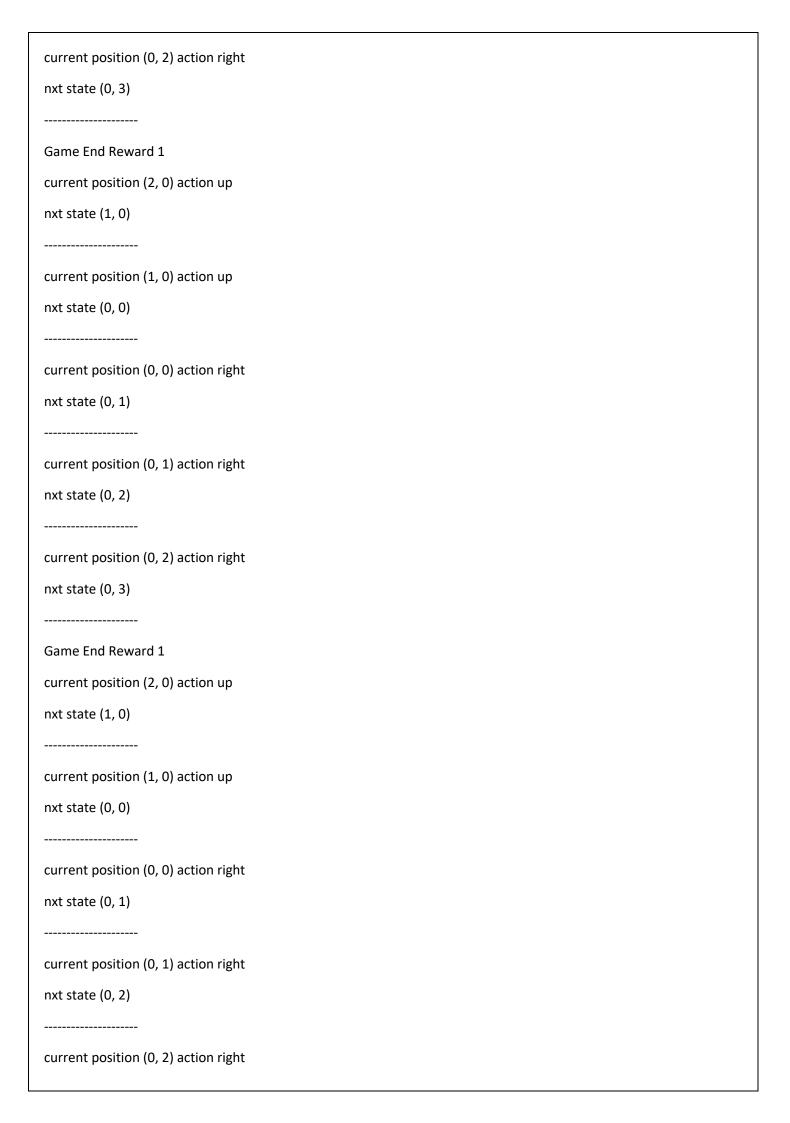






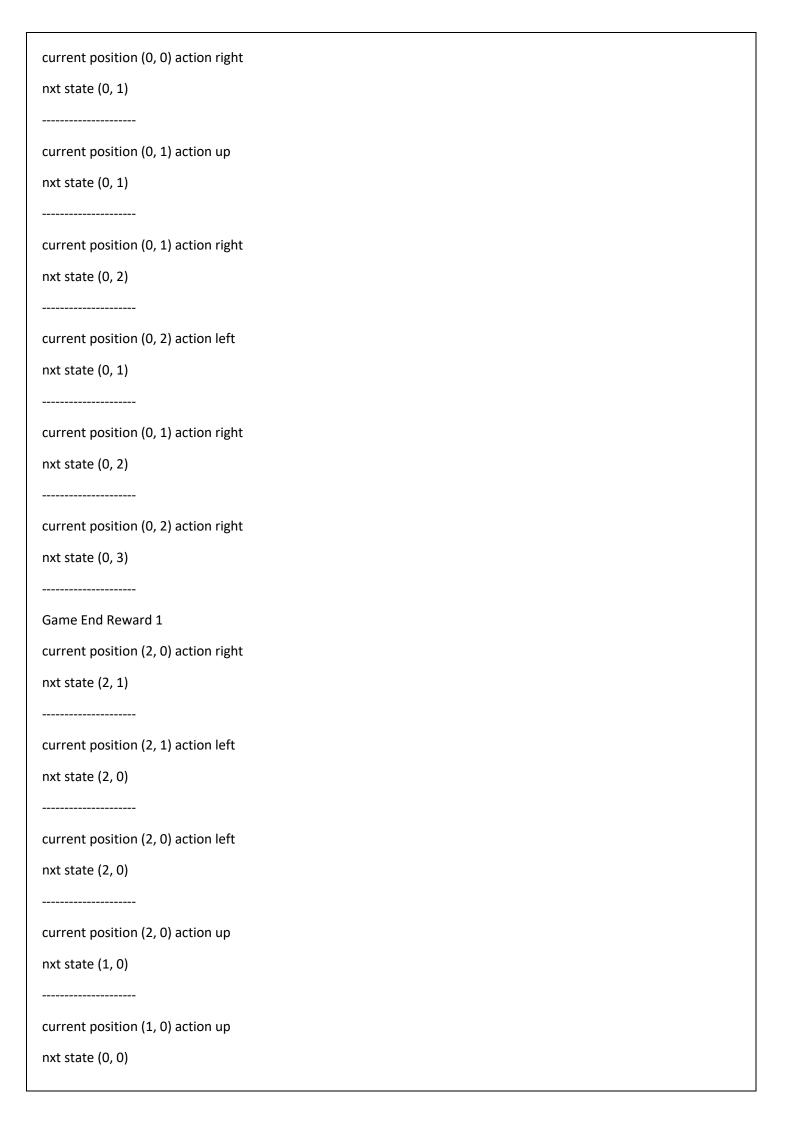


nxt state (0, 0)		
current position (0, 0) action right		
nxt state (0, 1)		
current position (0, 1) action right		
nxt state (0, 2)		
current position (0, 2) action right		
nxt state (0, 3)		
Game End Reward 1		
current position (2, 0) action up		
nxt state (1, 0)		
current position (1, 0) action right		
nxt state (1, 0)		
current position (1, 0) action up		
nxt state (0, 0)		
current position (0, 0) action right		
nxt state (0, 1)		
current position (0, 1) action right		
nxt state (0, 2)		
current position (0, 2) action left		
nxt state (0, 1)		
current position (0, 1) action right		
nxt state (0, 2)		



nxt state (0, 3)	
Game End Reward 1	
current position (2, 0) action up	
nxt state (1, 0)	
current position (1, 0) action up	
nxt state (0, 0)	
current position (0, 0) action right	
nxt state (0, 1)	
current position (0, 1) action right	
nxt state (0, 2)	
current position (0, 2) action right	
nxt state (0, 3)	
Game End Reward 1	
current position (2, 0) action up	
nxt state (1, 0)	
current position (1, 0) action up	
nxt state (0, 0)	
current position (0, 0) action right	
nxt state (0, 1)	
current position (0, 1) action right	
nxt state (0, 2)	
current position (0, 2) action right	
nxt state (0, 3)	







nxt state (0, 3)
Game End Reward 1
current position (2, 0) action left
nxt state (2, 0)
current position (2, 0) action up
nxt state (1, 0)
current position (1, 0) action up
nxt state (0, 0)

current position (0, 0) action right
nxt state (0, 1)
current position (0, 1) action up
nxt state (0, 1)
current position (0, 1) action right
nxt state (0, 2)

current position (0, 2) action right
nxt state (0, 3)

Game End Reward 1
nxt state (1, 0)
current position (1, 0) action up
nxt state (0, 0)
current position (0, 0) action right
nxt state (0, 1)



current position (0, 0) action right
nxt state (0, 1)
current position (0, 1) action right
nxt state (0, 2)
account position (0, 2) action left
current position (0, 2) action left
nxt state (0, 1)
current position (0, 1) action down
nxt state (0, 1)
current position (0, 1) action right
nxt state (0, 2)

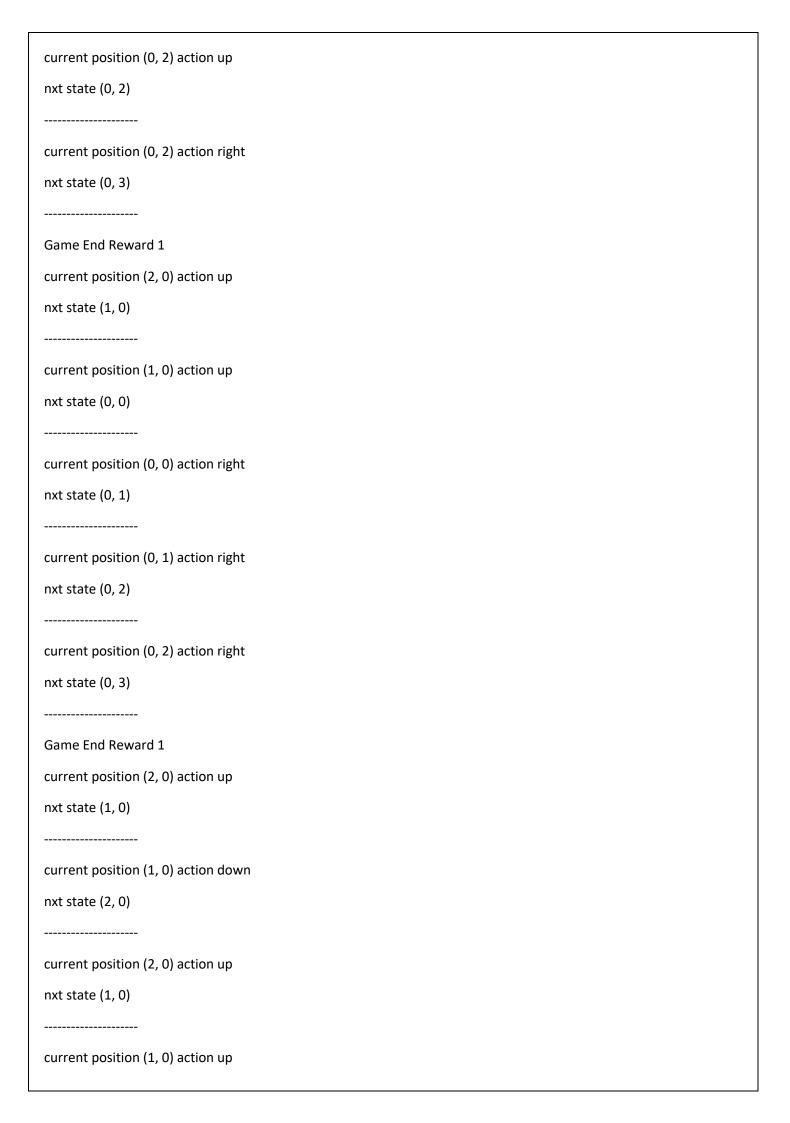
current position (0, 2) action right
nxt state (0, 3)
Game End Reward 1
current position (2, 0) action up
nxt state (1, 0)
current position (1, 0) action up
nxt state (0, 0)

current position (0, 0) action left
nxt state (0, 0)
current position (0, 0) action right
nxt state (0, 1)
current position (0, 1) action right
nxt state (0, 2)









nxt state (0, 0)		
current position (0, 0) action right		
nxt state (0, 1)		
current position (0, 1) action right		
nxt state (0, 2)		
current position (0, 2) action up		
nxt state (0, 2)		
current position (0, 2) action right		
nxt state (0, 3)		
Game End Reward 1		
current position (2, 0) action up nxt state (1, 0)		
current position (1, 0) action right		
nxt state (1, 0)		
current position (1, 0) action up		
nxt state (0, 0)		
current position (0, 0) action right		
nxt state (0, 1)		
current position (0, 1) action right		
nxt state (0, 2)		
current position (0, 2) action right		
nxt state (0, 3)		

Game End Reward 1
current position (2, 0) action right
nxt state (2, 1)
current position (2, 1) action left
nxt state (2, 0)
aurrent nesition (2, 0) estion up
current position (2, 0) action up
nxt state (1, 0)
current position (1, 0) action up
nxt state (0, 0)

current position (0, 0) action right
nxt state (0, 1)

current position (0, 1) action down
nxt state (0, 1)
current position (0, 1) action right
nxt state (0, 2)
current position (0, 2) action right
nxt state (0, 3)

Game End Reward 1

0.986 0.995 0.998 1.0
0.941 0 0.425 -1.0
0.827 0.453 0.03 -0.135

None
Result:
The result of running the code is the optimal state-value function for the gridworld. It shows that the agent has learned to reach the winning state with a high reward and avoid the losing state with a low reward. The code also demonstrates how the agent learns to navigate the gridworld through trial and error using Q-learning. Overall, the code is successful in implementing a basic RL algorithm and showing how it can be applied to a simple problem.