MEMORIAL UNIVERSITY OF NEWFOUNDLAND

DEPARTMENT OF MATHEMATICS AND STATISTICS

A7 **MATH 6205** DUE: APRIL 5TH, 2021

This is a bonus assignment, so you do not have to complete it! Upload your solution to Brightspace at the beginning of class on Monday April 5th, 2021.

1. Implement *value iteration* for general $n \times m$ gridworlds with an arbitrary but prescribed number of terminal states (and their associated rewards) and inaccessible states in `Python`. Your environment should be defined by specifying the number of rows n and columns m as well as a list of inaccessible states and terminal states with their associated values. All non-terminal transitions should incur a reward of -1.

   Test your implementation by solving the gridworlds given in Tables (1) and (2).

   Table 1: A 4×4 gridworld. Gray states are inaccessible, the blue states issues a reward of 0, all other transitions a reward of -1.

			🟦
	▨	▨	
			▨
			🟦

   Table 2: A 5×6 gridworld. Gray states are inaccessible, the green state issues a reward of 10, the blue state a reward of -10, all other transitions a reward of -1.

	▨				🟩
	▨				
	▨		▨	▨	▨
	▨				
					🟦

Notes: There are many ways how this can be done in `Python` but one elegant way would be to implement this using *object-oriented programming.* Here, we would design the environment as one class and the agent as another class. Class methods that the environment would have to provide include (i) issuing the next state based on a given action and (ii) issuing a reward based on the state the agent is in. The environment also keeps the value function for the entire gridworld as a class variable. Class methods the agent would have to provide include (i) an action method, which based on the current state the agent is in will decide which action to take next (based on greedy action selection) and (ii) value iteration over all states of the environment itself where the agent would cycle through all possible states of the environment and updates them according to the value iteration update rule.

The advantage of this object-oriented approach would be that you could easily recycle the classes for the environment and/or agent for other problems. Still, you do not have to use this object-oriented approach, and you can solve this problem any way you see fit.