# **Assignment 2 (10%)**

**Implementing MDP and Reinforcement Learning for GridWorld**

DUE DATE: Tuesday, November 7th, 2023, 11:59 PM ET

**Objective:**

* To enhance the provided "GridWorld" game by integrating concepts of Markov Decision Processes (MDP) and reinforcement learning, aiming to enable the robot to learn optimal policies for navigating through the grid to reach the goal.

**Assignment Guidelines:**

* All students in this course must read and meet the expectations described in the [Student Academic Integrity](https://intranet.laurentian.ca/policies/2017.09.19%20-%20Policy%20and%20Procedures%20on%20Academic%20Integrity%20-%20EN.pdf).
* Assignments must follow the programming standards document published on the course website in the D2L. Marks will be taken off if standards are not followed.
* **Submit just .py files (you can implement the algorithms into two separate .py) AND the associate report in PDF per group**. Name the file based on your group “ID” and the assignment number, exactly as in this example for **assignment 2 and** **group 1**: **COSC\_4117EL\_A2\_G1.py**. Same naming convention applies to the PDF, **COSC\_4117EL\_A2\_G1.pdf**.
* **Do NOT zip the files** that you submit.
* You may submit the assignment multiple times, but only the most recent version will be marked.
* After the due date and time, a late penalty of 2% per hour, or a portion of an hour, will be applied. After 49 hours, the penalty is 100% and submissions will no longer be accepted. The date and time of the last file submitted control the mark for the entire assignment.
* These assignments are your chance to learn the material for the exams. **Code your assignments independently (or solely within your group)**. We use software to compare all submitted assignments with one another, and pursue academic dishonesty vigorously. **You must complete the Honesty Declaration in the D2L before you will be able to submit your assignment.**

**Introduction:**

Assume there is a grid world where a robot agent moves through a maze to find a goal. The world contains walls, traps, and gold. The robot gets rewards for collecting gold and penalties for stepping on traps. It also gets a penalty (living reward) for each step it takes to encourage finding the optimal path. In this assignment, you need to implement MDP AND Reinforcement Learning for the [GridWorld](https://drive.google.com/file/d/1thZhVN-KcsbBGJHM61jAOxv56vXrKdFW/view?usp=sharing) provided as described in PART 1 and PART 2.

**PART 1: MDP (Markov Decision Process) [20 marks]**

1. State and Action Space:
   1. Definition and implementation of the state space (each cell in the grid) and action space (possible movements like up, down, left, right).
2. Transition Probabilities and Reward Function:
   1. Determine the transition probabilities, i.e., the likelihood of transitioning from one state to another given an action. You need to introduce stochasticity (noise = 0.2) where there's a chance the robot doesn't move in the intended direction.
   2. Define the reward function for GridWorld. Rewards can be obtained from gold cells (+10), penalties from traps (-10), and a living reward (-1) for standard moves to encourage the robot to find the shortest path. Assume discount factor gamma, γ = 0.9.
3. Value Iteration Algorithm:
   1. Implement the Value Iteration algorithm to compute the value function for each state in the grid.
   2. Initialize values for all states (cells) in the grid, typically to zero.
   3. Iteratively update the value of each state based on the *Bellman equation* until the value function converges (i.e., changes between iterations are below a small threshold, e.g. 0.0001).
4. Visualization:
   1. Once the value function is computed and has converged, derive an optimal policy by selecting the action that maximizes expected value for each state.
   2. Move the robot based on the chosen action. You need to **add a delay after each action** to allow the user to observe the robot's movements and continue until the robot reaches the goal state.

*Note: some of the features may already be implemented in the grid world domain.*

P**ART 2: Reinforcement Learning Algorithm [30 marks]**

1. Q-Learning:
   1. Initialize a Q-table with all zeros.
   2. Implement the Q-learning algorithm with exploration (using **ε-greedy strategy**) to find the best policy for navigating the grid world.
   3. Update the Q-values based on rewards obtained and the Q-learning update rule.
   4. Let the robot **explore** the grid world multiple times (episodes) to learn the optimal policy.
2. Experiment and Report:
   1. Adjust parameters such as learning rate (α) and discount factor (γ) to observe variations in learning.
   2. Analysis on the convergence of the algorithm: How many episodes did it take for the robot to learn an optimal or near-optimal policy?
   3. compare and contrast the performance of MDP (Value Iteration) and Q-Learning: convergence and optimal policy extraction (which algorithm scores higher based on the chosen parameters).
3. Visualization:
   1. After convergence with the optimal policy, move the robot based on the chosen action. Again, add a **delay after each action** to allow the user to observe the robot's movements and continue until the robot reaches the goal state.

**Evaluation:**

Your implementation will be evaluated based on:

* Correctness of the MDP formulation.
* Successful implementation and convergence of the policy iteration and Q-learning algorithms.
* Quality of visualization and insights from the learned Q-values.

**Tips:**

1. Make sure to handle edge cases where the robot is near walls or corners of the grid.
2. Use a diminishing ε in the ε-greedy strategy to balance between exploration and exploitation during learning.

**Submission:**

1. Submit the modified code with your implementations as .py files.
2. You may submit more than one .py file for different algorithms. Make sure you rename the filename accordingly.
3. Submit your report as a PDF file, include a detailed report explaining your approach, any challenges faced, and the decisions made. Include visualizations, tables, or charts that support your findings.
4. Your assignment should be self-contained, meaning a person should be able to understand your process and results just by reading your report and going through your code.

**Note on Group Contributions and Grading:**

If any group member believes that another member of their group deserves a lower grade due to their contribution level, they are encouraged to address this concern. To formalize this, the group can include an additional section in their report detailing the situation and the proposed grade adjustment, with the consent of all group members. It's essential that all group members agree and provide their consent for any proposed grade changes.

It's always best to communicate openly within your group and seek collaborative solutions. However, if discrepancies in contributions are significant and consensus is achieved, this mechanism ensures fairness in grading.

**Best of luck! Remember, the process and learning are as important as the final results.**