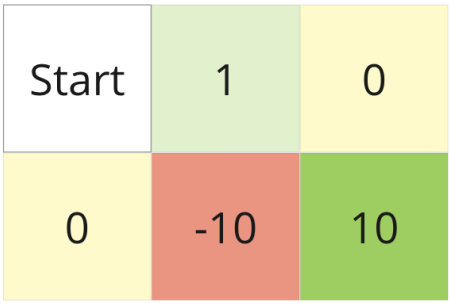
LAB 4

Task

Given the sample environment below, implement a Q-learning function to update corresponding the Q-values. At any of the states (6) actions (4): *Up, Down, Right, Left* can be performed. *The second page of this document includes some useful information to aid with this.*



Deliverables & Instructions

1. A Jupyter / Colab notebook showing your implementation(s). Include comments to explain your code.
    a. The final cell in your notebook should have an output of your final Q-Values from your Q-Table after training is complete.
    b. Rename your notebook to this format: **Group - StudentNumber.ipynb** e.g. A-111111111.ipynb OR B-11111111.ipynb OR C-111111111.ipynb
    c. Upload all files on E-Learning. Do not send any files via email as these will not be graded. If E-Learning fails, an alternative Dropbox link will be shared.

2. Any form of plagiarism e.g *(but not limited to)* Copying work from Github, Using AI tools (*ChatGPT, Bing Chat, Copilot,* etc.) will automatically lead to a zero score

Since this is the same example discussed in class (2 timesteps), you should follow this order of operations to better complete this lab:

1. Define an environment (store rewards) as some 2-dimensional structure. For simplicity store the Start state as **-10**
2. Define a Q-table to store pairings of *states-action & q-values*.
3. Define a Q-learning algorithm. For simplicity, you can use the same logic demonstrated during the physical classes last week and stick to these values for hyperparameter:
    a. Learning Rate = **0.1**
    b. Gamma (Discount Factor) = **0.9**
    c. Epsilon = 0.1
4. Iterate and update q-values for 20,000 iterations (episodes). Optional: *To avoid never-ending episodes, implement a tracker which terminates episodes where 4 or more moves don't lead to getting to the goal (10 - reward)*
5. You can do all this without defining a single function however that may mean difficulty in debugging in case you run into errors