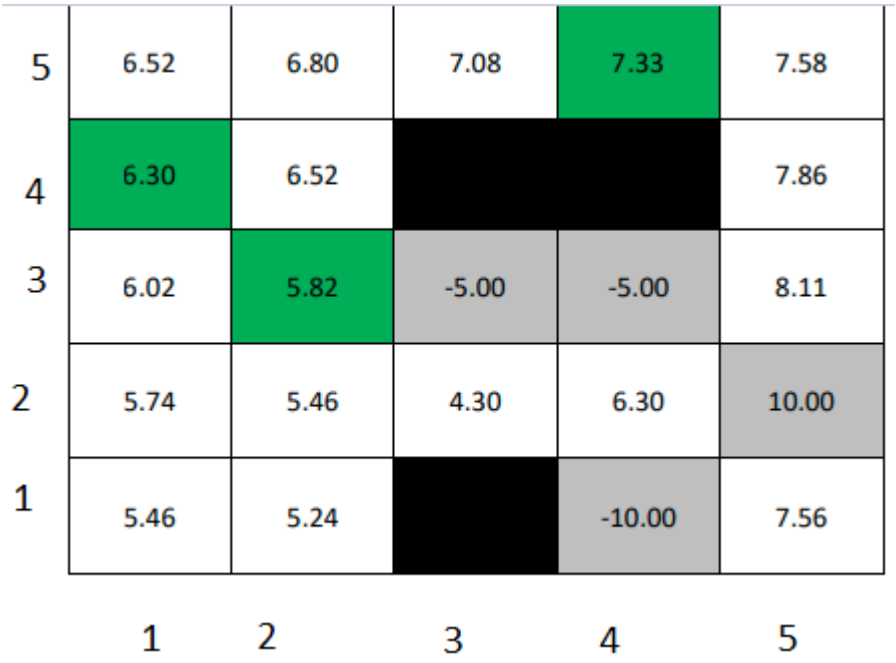
Portfolio 3-Reinforcement Learning

MDP: Only present matters, the world is stationary (rules don't change)



Definitions

States: Combination of Coordinates, e.g. (5,2) = "10.00"

Actions: Up, Down, Left and Right.

Model: Pr (s' | s, a) [Rules of the world]

0.8% for chosen action, 0.1% for failure **per** perpendicular.

E.g.

Chosen action = up (0.8):

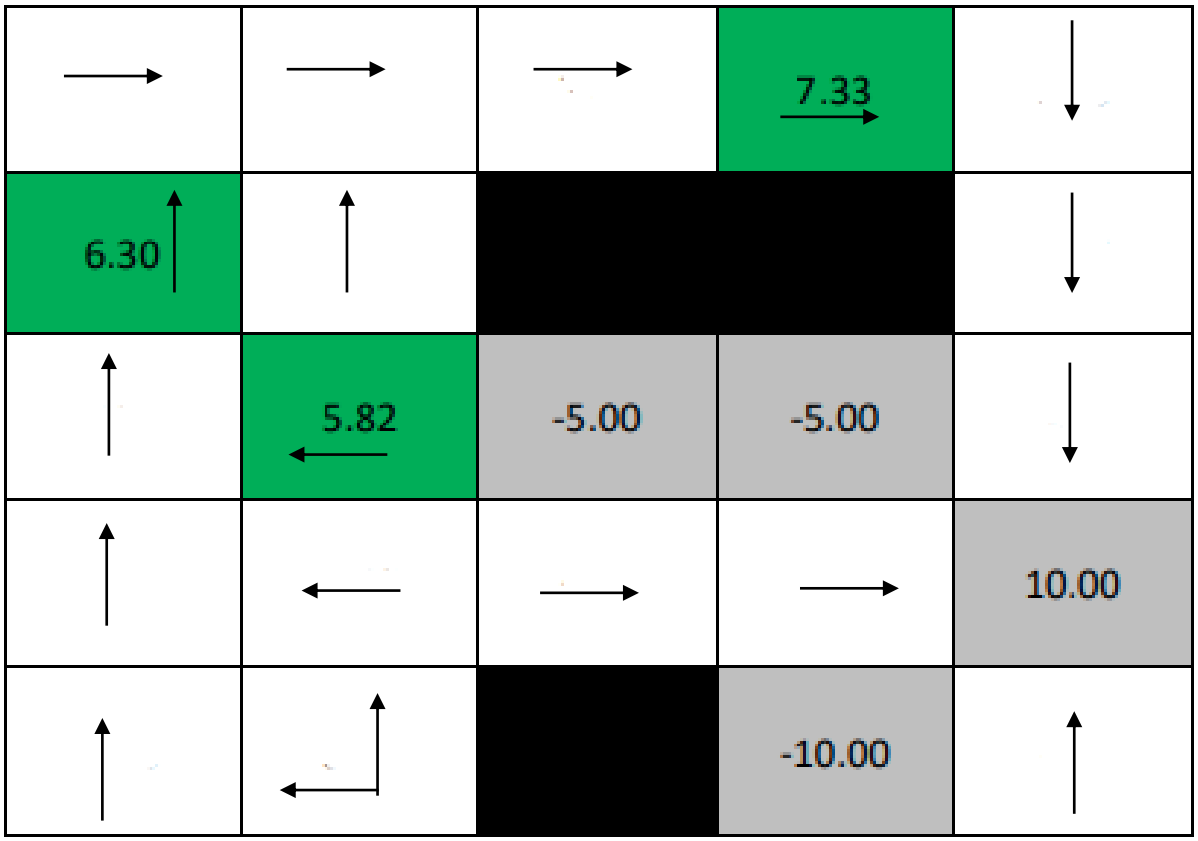
Failure = left (0.1), right (0.1):

Reward: receives a reward of -0.2 in a non-terminal state or of the value indicated below if in a terminal state

Policy: π(s) -> a: Given a state return an optimal action

π* is the optimal policy that maximized the long-term optimal reward.

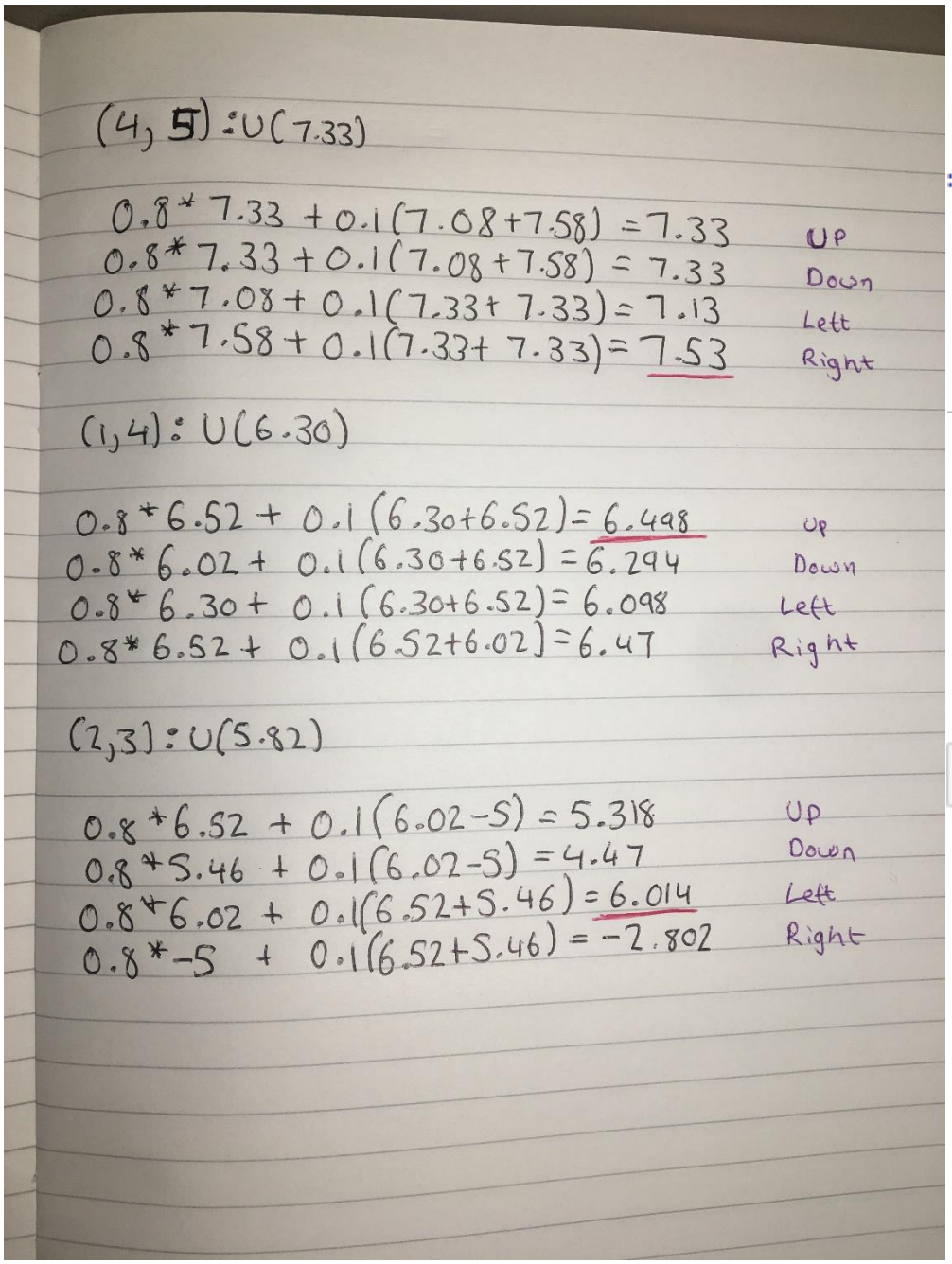
Subtask 3.1



Calculator:

```
In [1]: def optUtilCalc(up,down,left, right):
            util =[]
            upUtil = ((0.8*up) + (0.1*(left+right)))
            downUtil = ((0.8*down) + (0.1*(left+right)))
            leftUtil = ((0.8*left) + (0.1*(up+down)))
            rightUtil = ((0.8*right) + (0.1*(up+down)))
            print("Up: ", upUtil)
            print("Down: ", downUtil)
            print("Left: ", leftUtil)
            print("Right: ", rightUtil)
            util.append(upUtil)
            util.append(downUtil)
            util.append(leftUtil)
            util.append(rightUtil)
            util.sort()
            print("Max: ", util[-1])
```

```
In [12]: optUtilCalc(6.52,5.46,6.02,-5)
```




```
optUtilCalc(7.33,7.33,7.08,7.58)
```

```
Up:  7.330000000000001
Down:  7.330000000000001
Left:  7.130000000000001
Right:  7.53
Max:  7.53
```

```
optUtilCalc(6.52,6.02,6.30,6.52)
```

```
Up:  6.498
Down:  6.098
Left:  6.2940000000000005
Right:  6.470000000000001
Max:  6.498
```

```
optUtilCalc(6.52,5.46,6.02,-5)
```

```
Up:  5.3180000000000005
Down:  4.470000000000001
Left:  6.014
Right:  -2.8019999999999996
Max:  6.014
```

Justification

- If we know the Utility/value for each state, it is possible to infer the optimal policy.
- $\pi*$ can be inferred directly from utility

Can be found by finding the action which results in the highest expected utility from state s to s', or written simply, an optimal policy is one which results in an optimal value function.

Subtask 3.2

TDL is passive as it observes a temporal sequence of input states that works towards a final reward signal over some trials and experiments, interacting with the world to find out for itself using some fixed policy.

In this example we do not have a fixed policy and instead we need to decide what to do as there is no fixed policy it can act upon, as we do not have access to probabilistic transitions, we cannot use the bellman equation for choosing a an action in a given state, “not enough information to act upon”. And so, active learning is needed to learn and then act upon an optimal policy.

In an active learning scenario, we can generate Q value allows for action selection whereas utilities in a passive TDL do not (with the absence of probabilistic transitions); therefore, we would know what state we would end up in given some particular action.

Q-Learning so would be happy to take more risks hitting large negative rewards in training but would converge to an optimal policy; regardless of the current agent policy. SARSA on the other hand is more risk averse due to an on-policy algorithm, which may cause issues with convergence.

Q-Learning would be the choice.