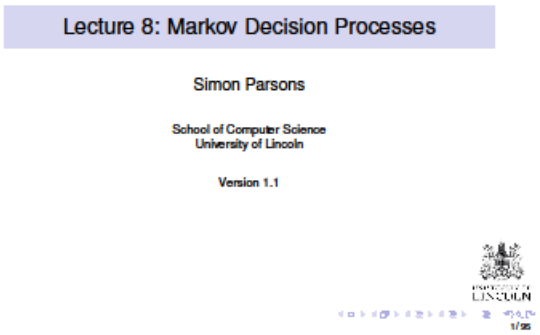
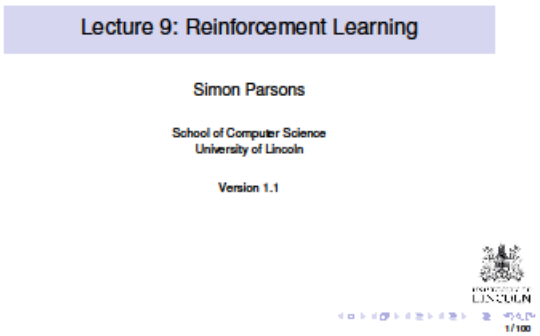
The Code Explanation and the Link for the reference that I used.

The concepts that are taught in the lecture 08 and lecture 09 proved really helpful for me to develop my code.





As in the abstract given for the assessment, I have coded in the Tallon part for three conditions

a. Control code for Tallon to navigate the arena for as long as possible when the dungeon is fully observable.
b. In this the Tallon has to navigate the arena when the dungeon is partially observable.
c. In this case, the performance should be observed by changing the number of pits, meanies, bonus stations, and speed.

In this, I have used Markov Decision Process (MDP) as it is used to describe an environment in reinforcement learning. Only the tallon part has to be coded. For part-c changing the size of the arena, the number of the pit, visibility, and other configurations the config code is changed.

In the case of fully observable the tallon has the full knowledge about the environment and also has the time to react whenever it is near meanies or pit. But in the partially observable condition, the tallon does not know the environment and has less time to react.

First the other codes are imported to make the tallon to move and to make use other configuration and the environment. The reward points are given so that whenever tallon moves to a white cell, pit, and meanies the reward points are provided.

After that probability and the reward are checked whether they are well formed or not.

The mdptool box is then used to run the value iteration which is used to determine the position of the tallon in the grid.

The probability and the reward arrays are created. And the arrays are created to store the location of meanies, pits and bonus.

https://stackoverflow.com/questions/12141150/from-list-of-integers-get-number-closest-to-a-given-value.

I referred stackoverflow the above link to find the closest bonus, meanies, and pit to the tallon. In this lambda is used for finding the closest ones.

In the above link they have explained about finding the closest number and the solutions for it this helped me to find the closest meanies, pits and bonus that are close to the tallon.

```
p_p = lambda x: np.ravel_multi_index(x,self.grid_size) # Pit position
m_p = lambda x: np.ravel_multi_index(x,self.grid_size) # Meanies position
b_p = lambda x: np.ravel_multi_index(x,self.grid_size) # Bonus position


#https://stackoverflow.com/questions/12141150/from-list-of-integers-get-number-closest-to-a-given-value

# This part is to determine the closest bonus to tallon so that it can be printed in the output window:


for bonus in range(len(self.bonuses)):
    a_b.append(b_p((self.bonuses[bonus].y,self.bonuses[bonus].x)))
    # closest bonus to tallon
    c_b = min(a_b,key=lambda x:abs(x-tallon_position_in_grid))
    c_b = str(c_b)
    c_b = (int(c_b[0]),int(c_b[1])) if (len(c_b)>1) else (0,int(c_b))
    currentbonus = c_b
print("The Bonus that is  closest to Tallon is: ",c_b)
```

Figure 1: To find the closest

The main objective is to find the closest bonus and to avoid the closest meanies and pits by the tallon this is achieved with the help of the above link that I used to build my code.

Probability for each action is determined so that the tallon moves in any one direction.

In part-A as mentioned above the control code is created for the fully observable condition and makes the Tallon move as long as possible. The fully observable condition is set by changing the partial visibility to false in the configuration condition. And the output is taken for that.

```
# If partialVisibility is True, Tallon will only see part of the
# environment.
partialVisibility = False
```

Figure 2: Visibility

Part-B: For the partially observable condition the partial visibility is set to true so that the tallons movements are restricted. And the outputs are taken for that.

```
# If partialVisibility is True, Tallon will only see part of the
# environment.
partialVisibility = True
```

Figure 3: Visibility

Part-C: For this the configuration code is used to change the number of pits, size of arena, speed of the tallon, and number on meanies spawning.

```
# Dimensions in terms of the numbers of rows and columns
worldLength = 10
worldBreadth = 10

# Features
numberOfMeanies = 1 # How many we start with
numberOfPits = 3
numberOfBonuses = 2

# Control dynamism
#
# If dynamic is True, then the Meanies will move.
dynamic = True

# Control observability
#
# If partialVisibility is True, Tallon will only see part of the
# environment.
partialVisibility = True
#
# The limits of visibility when visibility is partial
visibilityLimit =6

# Control determinism
#
# If nonDeterministic is True, Tallon's action model will be
# nonDeterministic.
nonDeterministic = True
#
# If Tallon is nondeterministic, probability that they carry out the
# intended action:
directionProbability = 0.95
```

Figure 4: Changing the configuration.