PART 1:

In this part I have used five different policies with five different seed numbers as 17,25,100,200,500. I have run the experiment for 100 times where each experiment runs 10000 episodes from this my best results are from the seeds 17 and 500. The results I will be listing the best two based on the mean goal steps. The 1st result is better than the 2nd.

Result 1:

The generated policy is:

```
*** Policy ***
[2 3 0 0 1 2 3 1 0 0 1 1 3 1 2 2 0 2 0 0 2 1 1 2 0 0 2 3 2 3 1 0 2 2 2 2
1
 0 2 0 1 0 2 1 1 2 0 1 2 3 0 3 1 1 3 0 0 1 2 3 1 1 3 3]
```

Expectation: Since this is not an ideal policy to reach the goal state, I expected the policy will not perform well but at least it will reach 50-100 episodes in the goal state

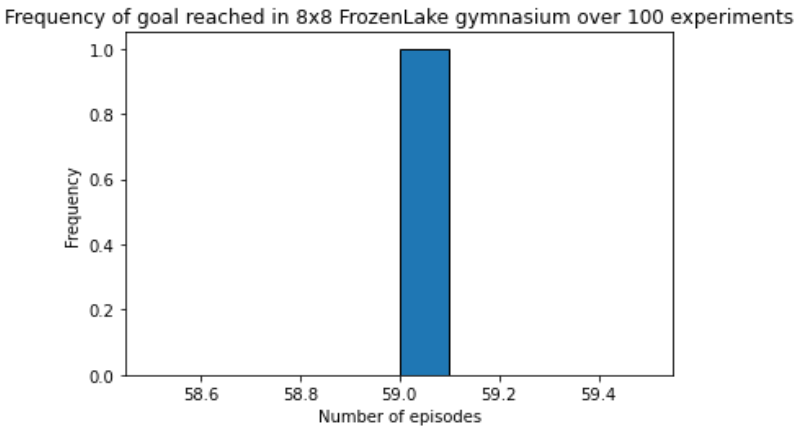
Result: In this policy I was able to reach the goal state only once and the rest of the time I ended up in the holes. This is because the policy is just to generate random action numbers for the given seed number.

The results I got are:

```
Mean successful runs: 1e-06
Standard deviation of successful runs: 0.0009999994999998747
Average number of steps that took to reach the goal: 59.0

*** RESULTS ***:
Goals:     1/1000000 =  1.000%
Holes: 999999/1000000 = 999999.000%
mean reward:          99.43000
mean goal steps:     58.00
```

Histogram plot:



Result 2:

The generated policy for this is:

```
*** Policy ***
[2 2 1 3 1 2 0 1 0 1 3 3 0 1 2 3 3 1 0 0 1 1 1 3 0 2 2 2 3 1 1 3 2 1 1 3 0
 0 3 1 1 1 2 2 2 1 3 1 2 0 0 0 3 1 0 1 1 1 1 2 1 3 2 3]
```

Expectation: This was my last policy for this part that is the fifth policy generated. I was increasing the seed number but there was no improvement in the results before this policy, so I expect this to not perform better than the before.

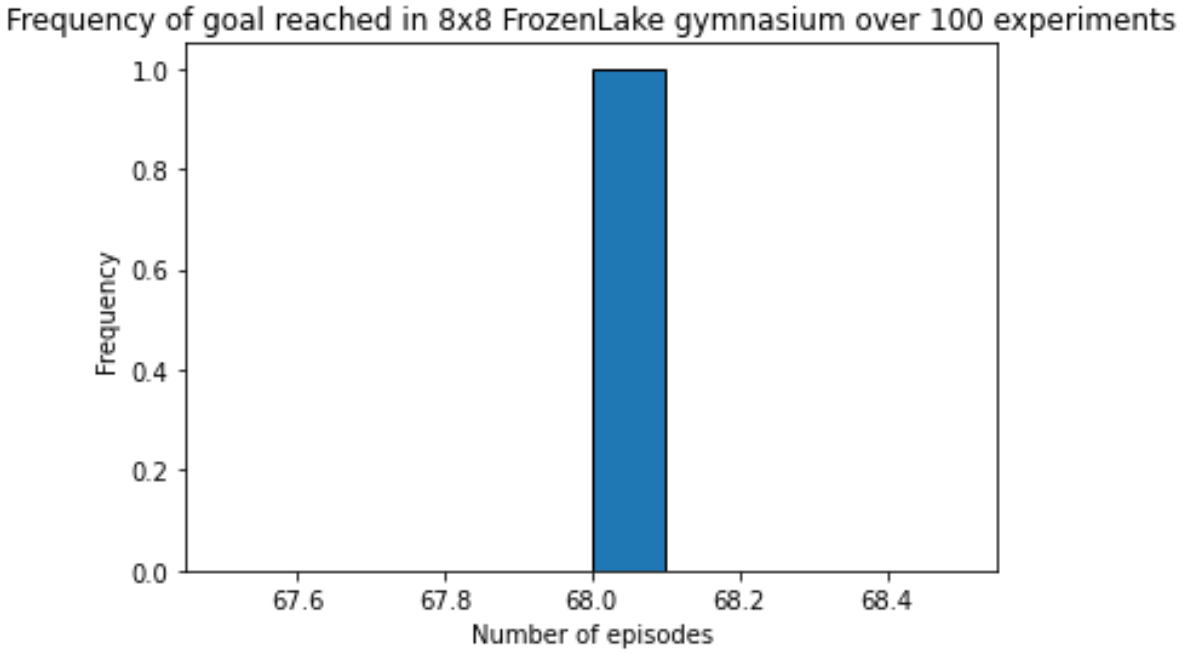
Result: It did perform better than the previous policies. But the result was same as the first with seed 17 except the mean goal and other metrics varied but not the goal state. I reran the policies again because we know reinforcement learning are random so I tried to run again still the result was same. Maybe if we try running it more then it might perform better.

The results I got for this policy is:

```
Mean successful runs: 1e-06
Standard deviation of successful runs: 0.0009999994999998736
Average number of steps that took to reach the goal: 68.0

*** RESULTS ***:
Goals:     1/10000 =  1.000%
Holes: 999999/10000 = 999999.000%
mean reward:          99.34000
mean goal steps:     67.00
```

Histogram plot:



Part 2:

In this part I have created the value iteration function to get a new policy based on the value function and the discount factor which here was 1.0. But to know what happens I tried the discount factor in one of the policies. The rest of the four policies I used the gamma to be 1.0 that is the discount factor. In this I have changed the theta value for the policies but there was not a very significant difference. The results I will be listing the best two based on the mean goal steps. The 1st result is better than the 2nd.

Result 1:

The generated policy for this is:

```
*** Policy ***
[1 2 2 1 2 2 2 2 3 3 3 3 3 3 3 2 0 0 0 0 2 3 3 2 0 0 0 1 0 0 2 2 0 3 0 0 2
 1 3 2 0 0 0 1 3 0 0 2 0 0 1 0 0 0 0 2 0 1 0 0 1 2 1 0]
```

The gamma and theta for this model I was using were 1.0 and 1e-15 respectively.

Expectation: This is the last policy in this part I produced before that I was decreasing the theta values and gamma value so I decreased the theta so low to 1e-15. I expected this to perform better than the other policies.

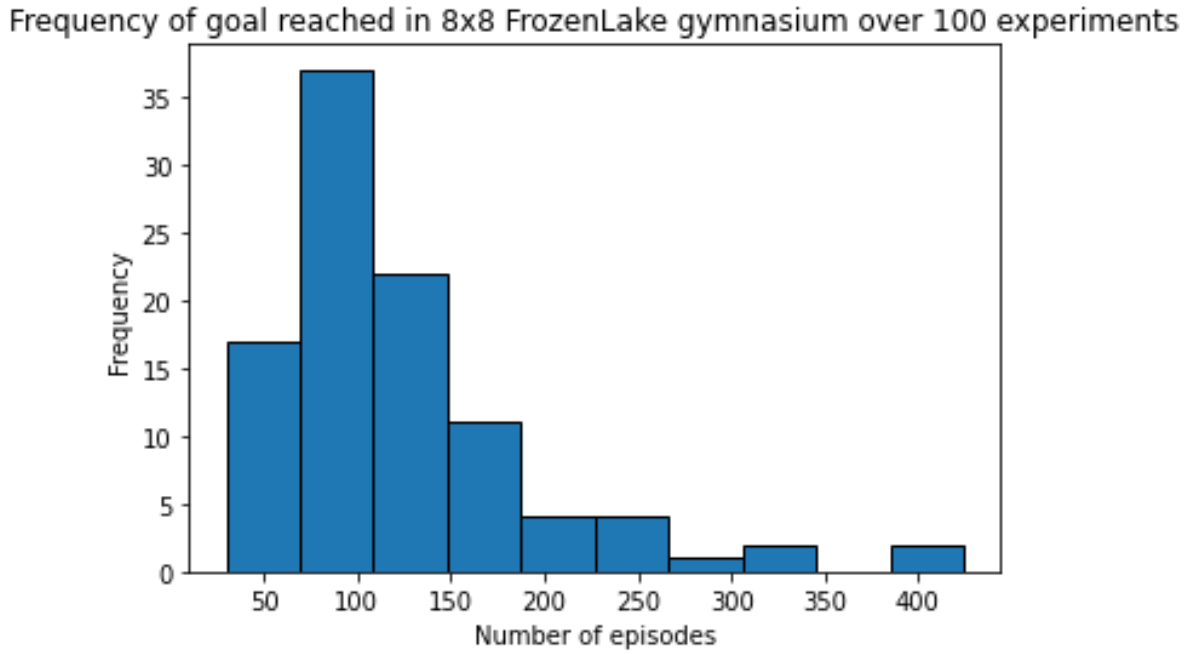
Result: The result was similar to what I expected but it is not a significant difference in the goals reached. The policy was significantly different from the other policies even the V(s) table.

The result I got is

```
*** Converged V(s) table ***
[1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         0.97820163
 0.92643052 0.         0.85661768 0.94623163 0.98207721 1.
 1.         0.9346049  0.80108992 0.47490377 0.6236214  0.
 0.94467761 1.         1.         0.82561308 0.54223433 0.
 0.53934275 0.61118923 0.85195561 1.         1.         0.
 0.         0.16804079 0.38321763 0.44226934 0.         1.
 1.         0.         0.19467347 0.12090475 0.         0.33240114
 0.         1.         1.         0.73155782 0.46311564 0.
 0.27746705 0.5549341  0.77746705 0.        ]
Mean successful runs: 0.0001
Standard deviation of successful runs: 0.009999499987499386
Average number of steps that took to reach the goal: 122.81

*** RESULTS ***:
Goals:   100/10000 = 100.000%
Holes: 999900/10000 = 999900.000%
mean reward:          9879.19000
mean goal steps:     121.81
```

Histogram plot:



Result 2:

The generated policy for this is:

```
*** Policy ***
[3 2 2 2 2 2 2 2 3 3 3 3 3 3 3 2 0 0 0 0 2 3 3 2 0 0 0 1 0 0 2 2 0 3 0 0 2
 1 3 2 0 0 0 1 3 0 0 2 0 0 1 0 0 0 0 2 0 1 0 0 1 2 1 0]
```

Expectation: This was my second generated policy; I was changing the theta value to 1e-3 to 1e-4 and gamma is 1.0. Since it is not a big difference, I was expecting it to perform similar to the first policy.

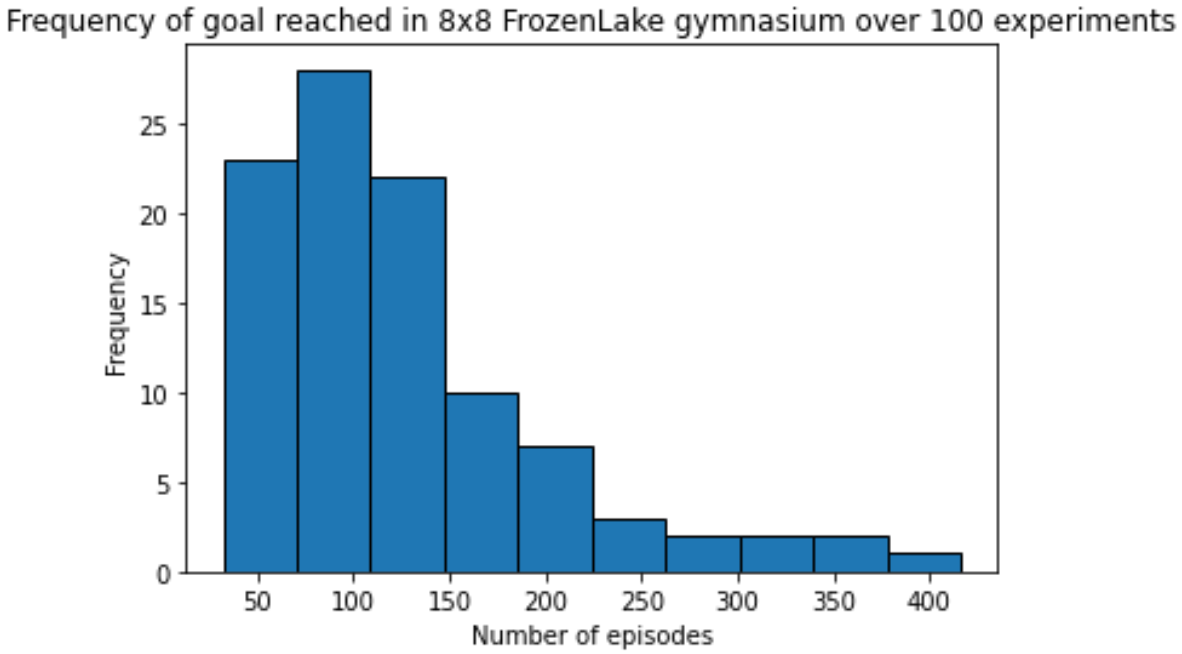
Result: As expected the result was quite similar but the mean goal step changed in this lesser than the previous but not that much still it was better than the previous so I am listing this in the top two.

The results I got are:

```
*** Converged V(s) table ***
[0.99936753 0.99939517 0.9994351  0.99947862 0.9995225  0.99956446
 0.99960198 0.99962943 0.99936064 0.99938168 0.99941627 0.99945753
 0.9995014  0.99954598 0.99959192 0.99964563 0.99883211 0.97711777
 0.92549914 0.         0.85614983 0.94578511 0.98167898 0.99967731
 0.99835343 0.93318584 0.80000329 0.47438821 0.62318331 0.
 0.94435755 0.9997231  0.99794439 0.82414646 0.54137107 0.
 0.53903075 0.6109269  0.85168455 0.99978099 0.99762189 0.
 0.         0.16788848 0.38299548 0.44207674 0.         0.99984845
 0.99739921 0.         0.19415037 0.12067642 0.         0.33231612
 0.         0.99992253 0.99728553 0.72949795 0.46179657 0.
 0.277437   0.5548753  0.777437   0.        ]
Mean successful runs: 0.0001
Standard deviation of successful runs: 0.009999499987499386
Average number of steps that took to reach the goal: 127.43

*** RESULTS ***:
Goals:   100/10000 = 100.000%
Holes: 999900/10000 = 999900.000%
mean reward:          9874.57000
mean goal steps:     126.43
```

Histogram plot:



Additional Explanations:

The performance of the optimal policy depends on the specific environment and the parameters used for the value iteration algorithm. In general, an optimal policy is expected to achieve the highest possible cumulative reward over time.

The optimal policy learned through value iteration may differ from the best fixed policy as it adapts to the dynamics of the environment and adjusts its actions based on the estimated value of each state-action pair. This allows the agent to explore different options and adapt to changes in the environment.

In terms of the RL agent's behaviour, if the agent has learned well, it should be able to navigate the environment effectively and reach the goal state with high probability. The agent's behaviour and learning can be observed by monitoring its actions and rewards over time.

In case the agent is not performing well, it could be due to the random policy that is being used, the agent is getting stuck in a loop or taking a long time to reach the goal state.

In any case, it's important to note that reinforcement learning is an iterative process, and the agent's performance may improve as it interacts with the environment and receives feedback on its actions.

The agent's actions becoming increasingly consistent and oriented towards the goal state as it learns from past experiences and updates its value function.

As it gains rewards for achieving the desired state and loses rewards for sinking into a hole, the agent's performance gradually gets better over time.

In order to determine the best path, the agent may potentially investigate other routes to the desired state.

Did it play well?

To answer this question yes it did because I was not expecting to see a huge difference between the random policy and value iteration policy this is because of the behaviour of the RL agent as I was saying it above. The agent learns from and updates the policy according to it.

General reflections:

Reinforcement learning is a new field I am learning and this assignment was really helpful for it and the difficulty was 3/5 because this is new to me. Particular difficulty is that if I made an error in the code the running time was really more so it was running for a while and reflected some errors to rectify that it took the assignment longer than I expected because the code was running long so I switched to google collab and used the GPU and after that it was that tough. I would start early next time.