# Module Implementations

## Environmental Simulation:

We created a map of open, closed and goal nodes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OPEN | OPEN | OPEN | OPEN | OPEN |
| OPEN | CLOSE | CLOSE | OPEN | OPEN |
| OPEN | CLOSE | GOAL | OPEN | OPEN |
| OPEN | CLOSE | CLOSE | OPEN | OPEN |
| OPEN | CLOSE | OPEN | OPEN | OPEN |
| OPEN | OPEN | OPEN | OPEN | OPEN |

With this map, we pick a random position for the agent, and have the agent navigation through the maze. Once it chooses a direction to move, the program will randomly see if the agent goes to the direction it wants or bounces to its current position. It will drift left and right at a rate of 30%, with 15% drifting left, and 15% drifting right.

## Ꜫ-Greedy:

We generate a random real number from [0,1], and we compare this number to Ꜫ. If it is higher, then the agent will pick the best direction to go to. If it generates a number less than or equal to Ꜫ, then it will randomly pick a direction to go to.

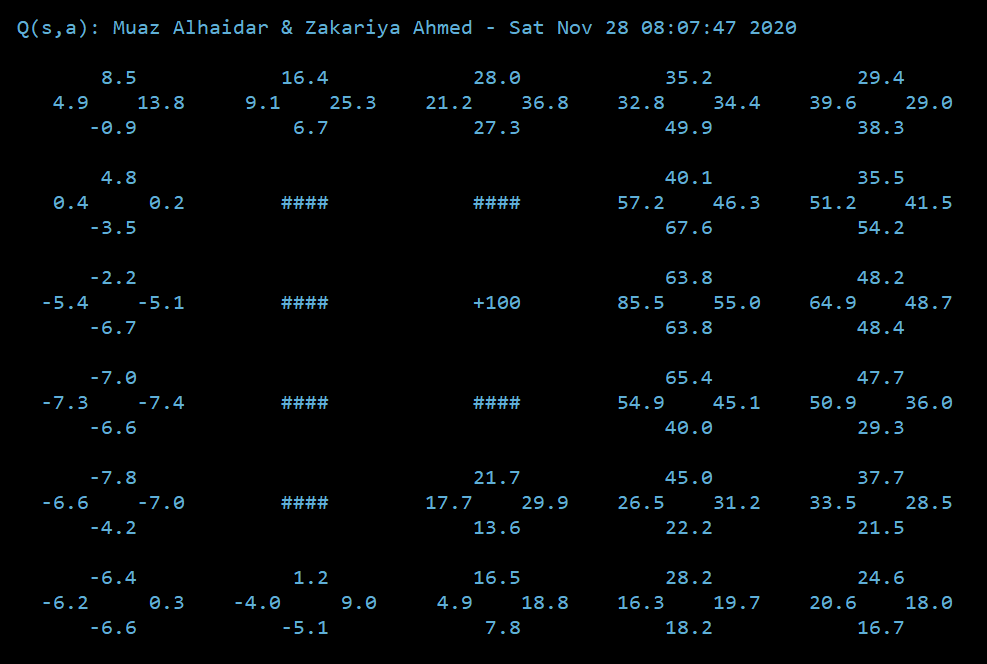
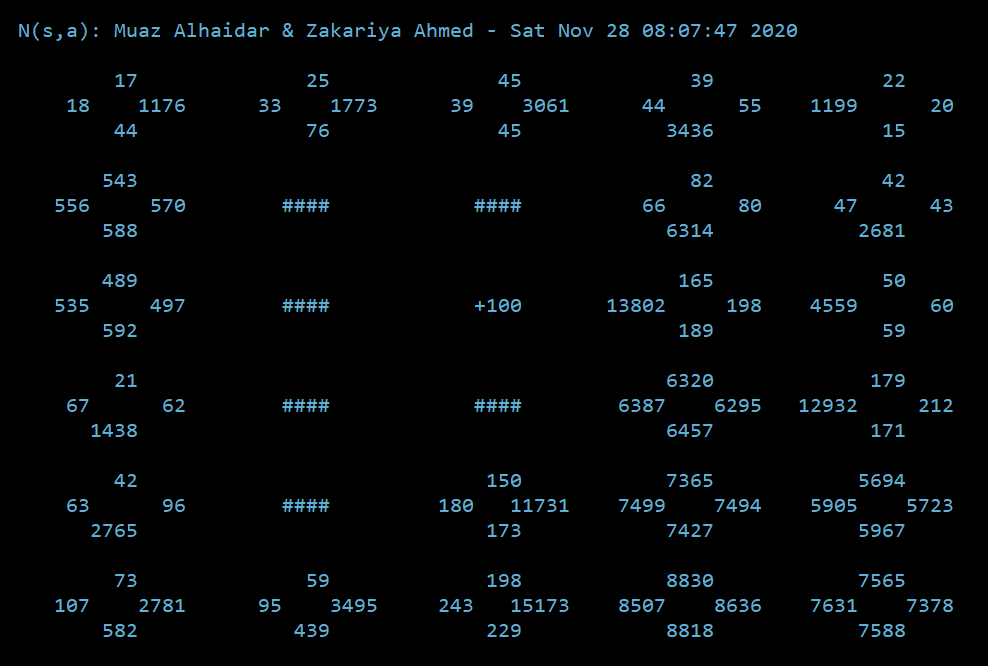
## Q-Learning

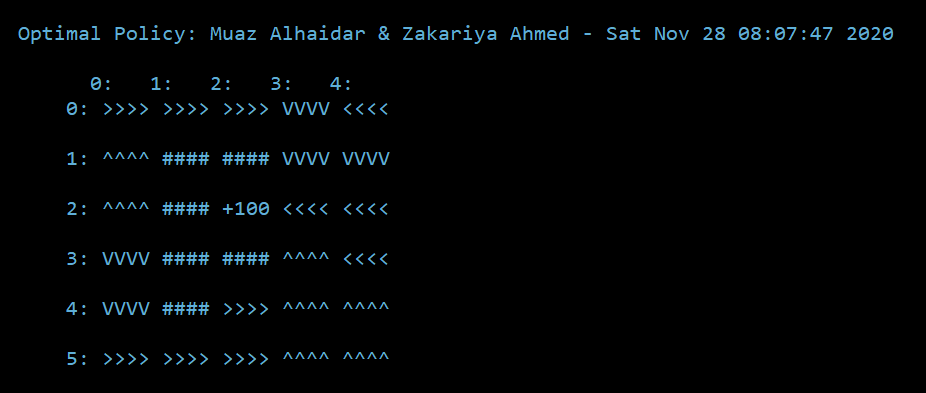
We implanted the function , by breaking each function in this expression to its own separate function.

R(s,a) returns the value of the position (zero for any node expect goal, 100 for goal position), finding the max Q value by iteration through all possible next steps the agent can make.

This concept stands for N(s,a), Q(s,a), and Equation 1 (incrementing N).

# Screenshots:





Team Contributions:

1. Zakariya Ahmed
   1. Design of Q-learning and implantation of printing grid and trials
2. Muaz Alhaidar
   1. Design of rest of program and implantation of Q-learning.