**Question 7**

We mentioned in the algorithm portion of Question 5 that when calculating our “c-value” (our new metric for judging which cell to plan to go to next, which is equal to the probability of finding the target at the cell / ((the Manhattan distance from the agent’s current position to this cell) + 1), we had initially planned to use A\* to try to plan paths so that the c-value could be determined with planned path length in the denominator as opposed to the Manhattan distance. The reason for this is that in specific cases, the Manhattan distance can be misleading (i.e., the agent must get around a very long wall despite only being 2 steps away from the destination cell). However, we determined that there was not a feasible way to have this planning happen when considering a new destination, even when filtered and optimized so we didn’t call A\* for every cell in the grid, because it drove up the runtime and there would be a great deal of extra computations that the program would have to execute.

In an ideal world though, to avoid situations where the Manhattan distance misleads our “c-value” calculation, we would have some system to be able to determine the actual distance very quickly between the current position and a potential destination. This would maybe result from extra processing power, divided workload via different threads/processes, a data structure to store the best-known distance from a cell to every other cell in the maze, etc. Obviously, not all these things are rapidly available or feasible for the average college student working off a 6-year-old laptop, but if we wanted to truly optimize Agent 8, being able to know the actual distances between a current position and a potential destination cell when updating the belief state could result in even more efficient costs.