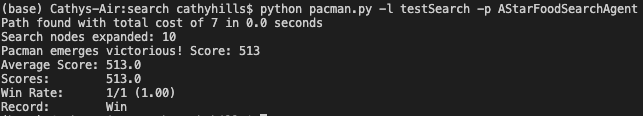
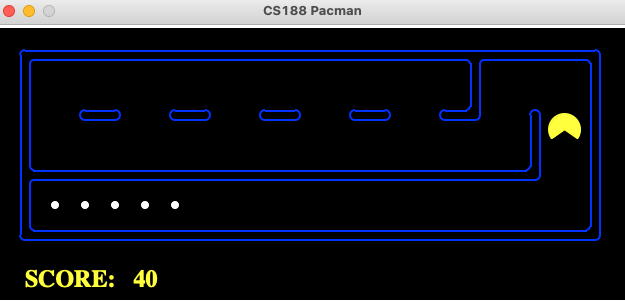
**PacMan Project - ReadMe - Week 6**

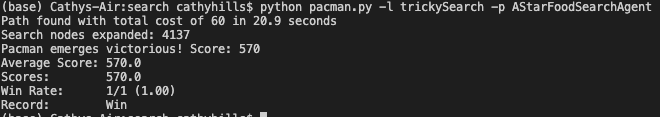
**Question 7:**

* **Summary**:
  + To complete this question, we build off of our already completed part 4 astar search algorithm. As FoodHeuritsic class is mostly already implemented, our goal is to implement a new admissible/consistent heuristic which can expand at most 9000 nodes when run on the tricky board game. To solve this problem, we first tried implementing our heuristic from the previous question, manhattan distance, but found better results when using mazedistance. Although the total running time may be slower due to multiple calls to bfs, the total nodes expanded is less.
* 1. **TestSearch** - python pacman.py -l testSearch -p AStarFoodSearchAgent

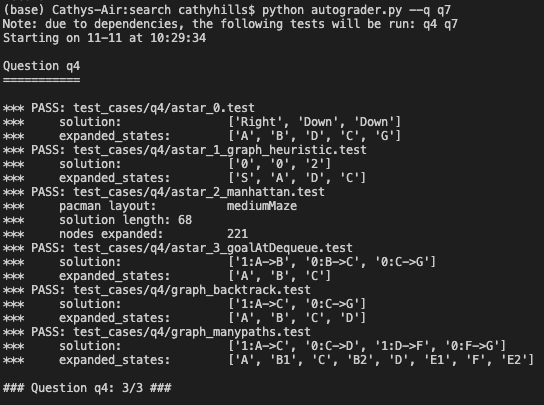


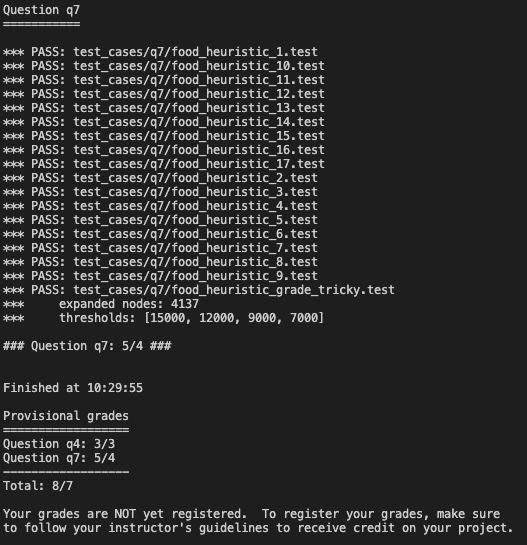
* 2. **TrickySearch** - python pacman.py -l trickySearch -p AStarFoodSearchAgent





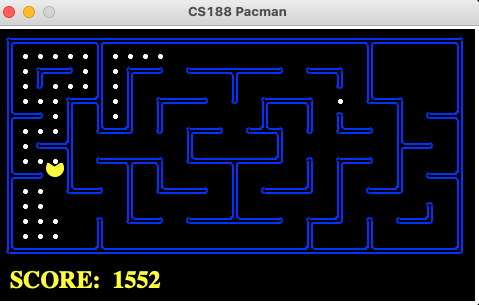
* 3. **AutoGrader** - python autograder.py --q q7



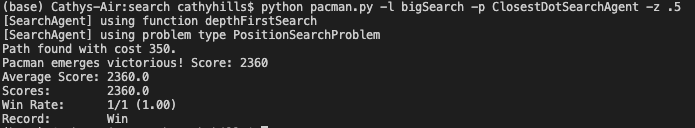


**Question 8:**

* **Summary**
  + Our goal for part 8 is to write an agent who will greedily get the closest dot. As given in the hint, we are to implement the goal state method within the anyfoodsearchproblem class, then we simply return a search previously implemented and pass it the problem object.
  + We observe that the shortest path is not always taken, but rather it will select the greedy choice based on where the food is. Repeatedly going to the closest dot will not always result in the shortest path, which can be illustrated by the screen below. Clearly, the lone dot on the left should have been eaten before traversing further on the left side.



* 1. **BigSearch** - python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5



* 2 - **AutoGrader** - python autograder.py --q q8

