Assignment 4
Trevor Lacoste

1)

```
### Update the evaluation function for the cell n: f(n) = h(n)
self.cells[new_pos[0]][new_pos[1]].f = self.cells[new_pos[0]][new_pos[1]].h
```

For the Greedy Best-First search algorithm, removed g(n) from the evaluation function.

A*

	g=1 h=17	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0		g=inf h=0
g=inf h=0	g=2 h=16	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0		g=inf h=0
g=inf h=0	g=3 h=15	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0		g=inf h=0
g=inf h=0	g=4 h=14	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0		g=inf h=0
g=inf h=0	g=5 h=13	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0		g=inf h=0
g=inf h=0	g=6 h=12	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0		g=inf h=0
g=inf h=0	g=7 h=11	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0		g=inf h=0
g=inf h=0	g=8 h=10	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0		g=inf h=0
g=inf h=0	g=9 h=9								g=inf h=0
g=inf h=0	g=10 h=8	g=11 h=7	g=12 h=6	g=13 h=5	g=14 h=4	g=15 h=3	g=16 h=2	g=17 h=1	g=18 h=0

Greedy Best-First

	g=1 h=17	g=2 h=16	g=3 h=15	g=4 h=14	g=5 h=13	g=6 h=12	g=7 h=11		g=inf h=0
g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=8 h=10		g=inf h=0
g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=9 h=9		g=inf h=0
g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=10 h=8		g=inf h=0
g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=11 h=7		g=inf h=0
g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=12 h=6		g=inf h=0
g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=inf h=0	g=13 h=5		g=inf h=0
g=inf h=0	g=20 h=10	g=19 h=9	g=18 h=8	g=17 h=7	g=16 h=6	g=15 h=5	g=14 h=4		g=inf h=0
g=inf h=0	g=21 h=9								g=inf h=0
g=inf h=0	g=22 h=8	g=23 h=7	g=24 h=6	g=25 h=5	g=26 h=4	g=27 h=3	g=28 h=2	g=29 h=1	g=30 h=0

In this scenario, the A* algorithm on the left is quicker at finding the goal node because it considers the actual cost and the estimated cost to the goal node. The greedy best-first search only goes to the node with the smallest cost. This approach causes the path for the greedy algorithm to be 30 steps, while the A* finds the goal node in only 18 steps, which is much quicker than the greedy approach.

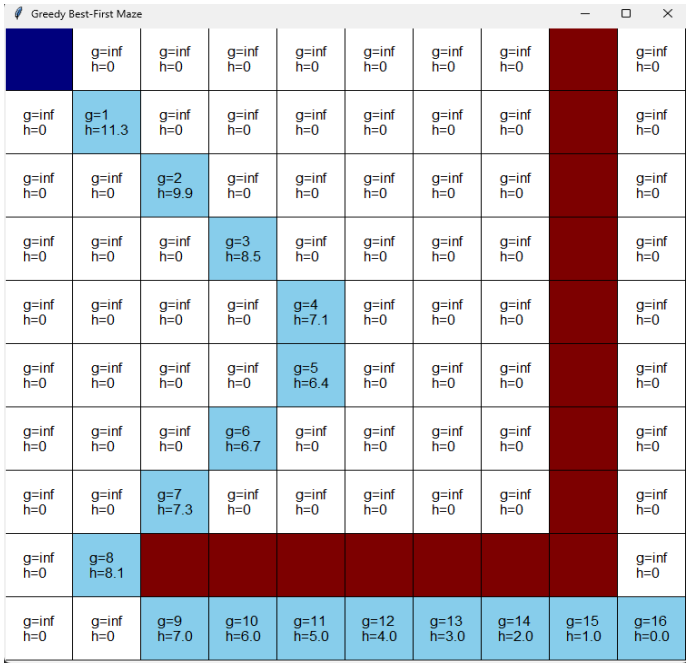
2)

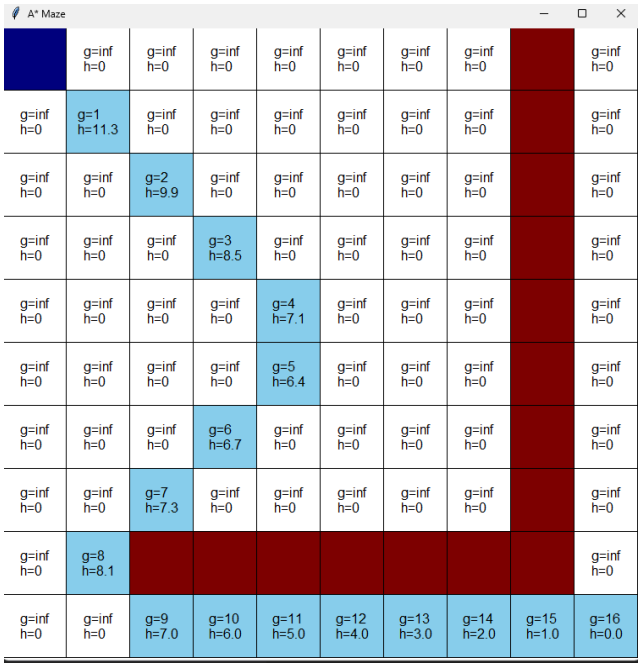
```
#### Agent goes E, W, N, S, NW, NE, SW, SE whenever possible
for dx, dy in [(0, 1), (0, -1), (1, 0), (-1, 0), (-1, 1), (1, 1), (-1, -1), (1, -1)]:
```

```
return round(math.sqrt(((pos[0] - self.goal_pos[0])**2) + ((pos[1] - self.goal_pos[1])**2)), 1)
```

Changed heuristic to calculate for euclidean distance instead of manhattan and allowed the agent to travel in diagonal directions now.

A* Greedy Best-First





After switching to Euclidean distance for the heuristic, the A* algorithm, and the greedy best-first algorithm both have the same path that finds the goal node in 16 steps. Switching heuristics had a minor effect on A* but made a huge difference in the efficiency of the greedy algorithm since the greedy approach wants to go directly to the goal node and moving diagonally makes that a more efficient path.

3)

1.

α	β	Observed Behaviour
Low	Low	With a low alpha and beta, the search behaviour is less efficient and may overlook a promising path and find a suboptimal solution.
Low	High	The algorithm focuses on the remaining cost to the goal node, which may cause the algorith to be faster but could also lead to an inefficient solution.
Equal	Equal	When the actual cost and heuristic are both

		important, the search behaviour is optimal and efficient as long as the heuristic is admissible.
High	Low	When the algorithm prioritizes minimizing the actual cost, the solution may be be optimal but it may take longer to find.
High	High	Since both actual cost and heuristic are given a high priority, the optimal solution can be found with good efficiency but it may be computationally costly.

2.

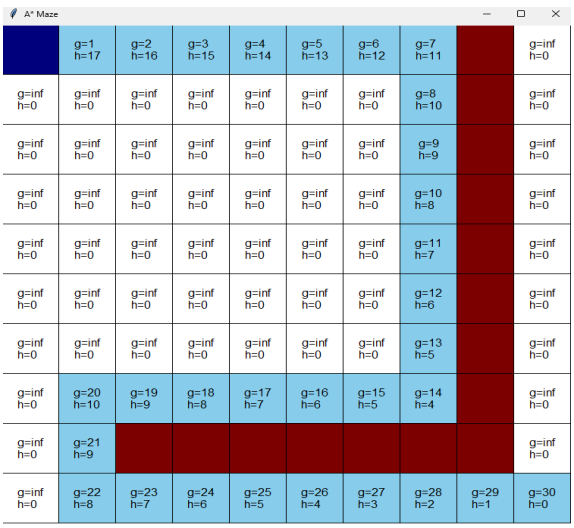
```
### Update the evaluation function for the cell n: f(n) = g(n) + beta * h(n)
self.cells[new_pos[0]][new_pos[1]].f = new_g + (2) * self.cells[new_pos[0]][new_pos[1]].h
```

β = **2**

β = **3**

β = **4**

β = 14



For the A* algorithm, as β is increased the algorithm eventually has the same path to the goal node as the greedy best-first algorithm. This is because the β value becomes so high that the priority of the heuristic far outweighs the priority of the actual cost. At first by barely increasing the beta from 2 to 3 then 4, the algorithm changes with each increase. After those small increases, it took changing the β value to 14 in order for another change to occur because using the actual cost led to a much better solution, but eventually the β was to great and ignores actual cost.

Conclusion:

Overall, the A* search algorithm is more reliable than the Greedy best-first search algorithm because it is complete and optimal so it will always find a solution. Greedy search can sometimes work well in finite spaces but it is not optimal and finding a path that is comparable to A* might only be possible with certain heuristics.