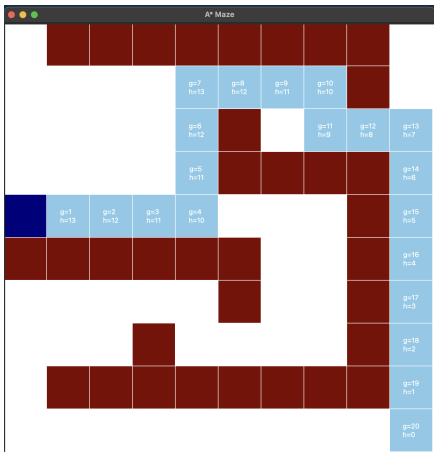
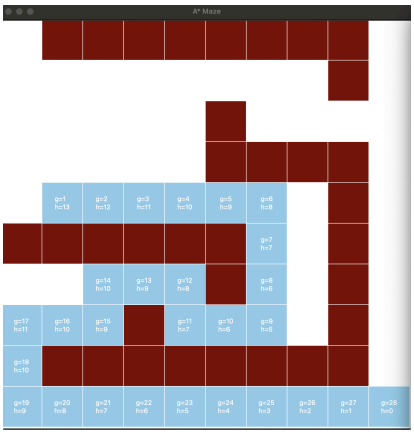
#HW3 problem 3 Simon Tice

a	b	Observed Behavior
a=1	b=1	This just multiplies both g and h by 1 so it just performs like the standard A* algorithm
a=1	b=0	This completely ignores the heuristic and just expands evenly from the start, putting all of the weight on the cost of the path. (Dijkstra's algorithm)
a=0	b=1	This puts the full weight of the algorithm on the heuristic and ignores the path cost so it acts as a greedy best-first algorithm.
a=1	0 < b < 1	This still acts very similar to the normal A* algorithm; however, it puts less weight on the heuristic and is more cautious of the actual cost. (less greedy)
a=1	b > 1	The heuristic dominates the calculation here and a lot less weight is put into the actual cost. It is still A* because it takes both into account but it is much more greedy. The higher b is, the greedier it is.
a > 1	b=1	The actual path cost dominates the calculation here and the heuristic has a weaker effect. This causes it to be a very conservative and nongreedy A* algorithm that only explores proven routes.
0 < a < 1	b=1	This still behaves very similar to the regular A* but it has more value on the heuristic than the path cost. This causes it to be a very slightly greedy version of A*

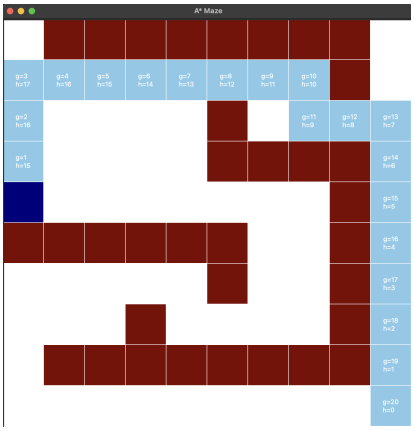


This is the initial optimal A* graph with no values associated with a or b (just as a reference for the other graphs).



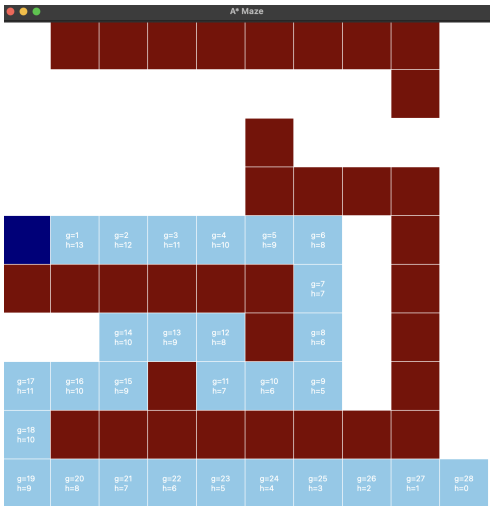
Here the value of b is set to 1 and the value of a is 0 which causes the algorithm to put all of the weight into the heuristic and not the actual cost causes it to be greedy and take a non-optimal path because it was Just going straight for the goal.

```
### Update the evaluation function for the cell n: f(n) = g(n) + h(n)

##### USE THIS FOR A*
a = 0
b = 1
self.cells[new_pos[0]][new_pos[1]].f = (new_g * a) + ((self.cells[new_pos[0]][new_pos[1]].h) * b)
```



Here the value of b is 0 and the value of a is 1 Which causes the algorithm to completely ignore the heuristic and just evenly expand from the start worrying about true cost only. This is why it goes straight up to start because it has no heuristic to go off of but it will still have an optimal path

```
### Update the evaluation function for the cell n: f(n) = g(n) + h(n)

##### USE THIS FOR A*
a = 1
b = 0
self.cells[new_pos[0]][new_pos[1]].f = (new_g * a) + ((self.cells[new_pos[0]][new_pos[1]].h) * b)
```



Here the b value is 25 and the a value is 1. Because the b value is so high it takes over the calculation and causes it to be heavily based on the heuristic, causing the actual cost to take a backseat making the algorithm very very greedy even though it still accounts for the actual cost.

```
### Update the evaluation function for the cell n: f(n) = g(n) + h(n)

##### USE THIS FOR A*
a = 1
b = 25
self.cells[new_pos[0]][new_pos[1]].f = (new_g * a) + ((self.cells[new_pos[0]][new_pos[1]].h) * b)
```