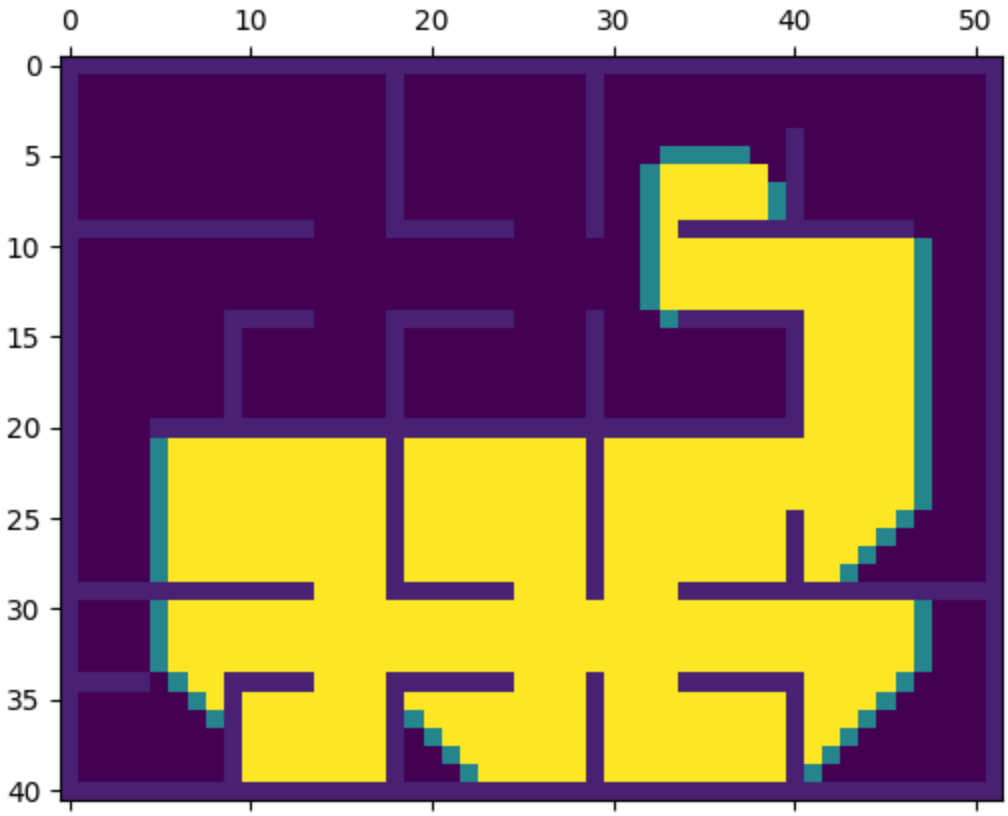
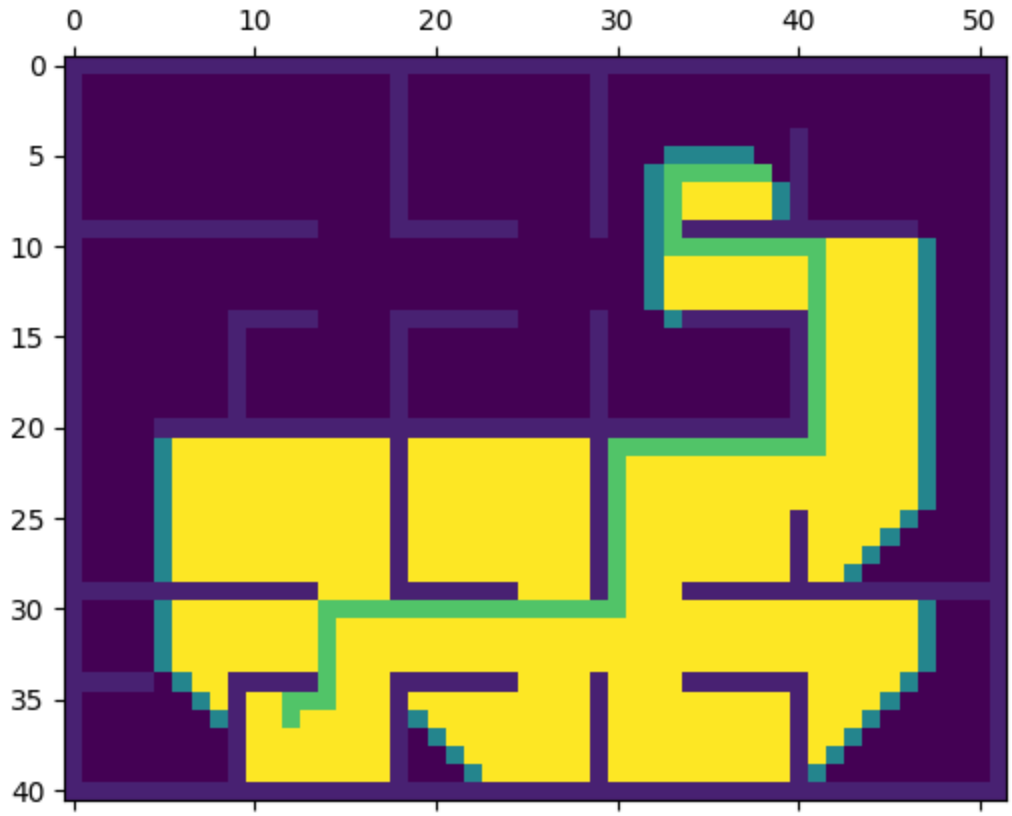
Project 3 Lab Report

1. Does your implementation work as expected? If not, what problems do you encounter?
    a. We encountered problems with the A* path planning algorithm as it looked more like djikstra's algorithm as we were doing it But when we switched to using our own logic it started to work as expected. We still printed out the open and closed list for the A* algorithm to look at how the algorithm searched through values and then used the closed list looking at it backwards to determine the shortest path. We started off parts 2 and 3 by figuring out how to move the black and yellow ball and how to make the end effector, but ran into trouble as the first keyboard inputs only changed the first two joints of the arm. We ran into issues trying to form the end effector from the transformation matrix since we only know what position we want and don't know velocity. We were unsure how to use the Jacobian, do we make the end effector velocities matrix given the joint angles of each angle? We realized too late that we were supposed to use the inverse Jacobian method to set each joint angular velocity to get an end effector velocity. We were unable to successfully implement parts 2 and 3 due to time constraints and the stressful circumstances around other classes and housing.
2. What did you learn?
    a. We learned about the A* algorithm and dijkstra's algorithm. We learned about nodes and heuristics and how to make a simple path using the

distance between the start (g) and the heuristic distance between the goal (h) to create a total cost for each node (f), making a path that moves along the nodes with the shortest cost. We used manhattan distance which was just $|x2 - x1| + |y2 - y1|$ and pythagorean distance $\sqrt{(x2 - x1)^2 + (y2 - y1)^2}$ and found manhattan distance to be more useful since we made the algorithm to only go left, right, up and down rather than in any direction. We learned about the Jacobian matrix and the Jacobian determinant. The Jacobian matrix is a matrix of the first order partial derivatives, which represent the velocities. The Jacobian relates the velocities of the end effector (x dot) and the joint velocities of each joint (q dot) with the function x dot = J * qdot. To get the joint velocities given the end effector velocities we use the inverse Jacobian to make the function qdot = J^-1 * xdot. The xdot is a 6x1 matrix containing the velocities of the end effector, the x, y, z and then the roll, pitch and yaw. $\overline{X} = [\overline{x}\ \overline{y}\ \overline{z}\ \overline{\alpha}\ \overline{\beta}\ \overline{\gamma}]^T$ The qdot is an nx1 matrix, where n is the number of joints containing the angular velocity of each joint in order. $\overline{q} = \left[\overline{q_1} \dots \overline{q_n}\right]^T$ Each column in the Jacobian matrix forms the end effector velocity given a change in the joint velocities. The jacobian takes the change in each q for qdot and derives it in terms of x, y, z, α, β, γ making a 6xn matrix, where n is the number of joints. For a DH table, a 4x4 transformation matrix represents the rotation and translation of the end effector, with the top left 3x3 matrix representing the rotation and the right left 3x1 matrix representing the translation.


0
10
20
30
40
50
0
5
10
15
20
25
30
35
40


0
10
20
30
40
50
0
5
10
15
20
25
30
35
40