1. What happens when you set the "gains" (source-of-error coefficients) of your controller too low? Describe your observations both for distance and bearing.

When we set the gains of our controller too low the distance and bearing calculations will be larger than expected. Further, if the gain becomes too large the controller will not consider the other eros and it will have trouble reaching the waypoint.

2. What happens when you set the distance threshold for your stopping criteria (selecting the next waypoint from your list) too low? What happens when you set it too high?

When we set the distance threshold too low the robot may miss the waypoint and when the threshold is too high the robot can undershoot the waypoint and not hit it as accurately as it should.

3. Describe how you implemented your feedback controller in Part 2.

We implemented the feedback controller by creating a method called feedback_controller(). In this method we have a if else statement that determines if the distance error is greater or less than .015. If the distance error is less than .015 then we set both the left and right wheel velocity to 0. If the distance error is greater than .015 then we enter an if else chain that determines the necessary speeds to reach our waypoint.

4. Does your implementation for Part 2 work? If not, describe the problems it has.

Our implementation for part 2 works as expected mostly. It bumps into a wall but is still able to make it to the waypoint in the kitchen.

5. What are the equations for the final controller from your implementation? What are your equations for θ'r *and* x'r in your feedback controller?

As follows

θ'r. If distance error > .04:

```
phi_l = (distance_error*distance_constant -
 (bearing_error*AXLE_LENGTH)/2)/AXEL_RADIUS

phi_r = (distance_error*distance_constant +
(bearing_error*AXLE_LENGTH)/2)/AXEL_RADIUS
```

x'r. If distance error <= .04:

```
phi_l = (distance_error - (heading_error*AXLE_LENGTH)/2)/AXEL_RADIUS
phi_r = (distance_error + (heading_error*AXLE_LENGTH)/2)/AXEL_RADIUS
```

6. What happens if you do not limit the wheel speeds to positive or negative MAX_SPEED? Think about what the robot will do in this case and what your odometry will predict.

If you do not limit the wheel speeds to positive or negative MAX_SPEED the controller code will miss-calculate the odometry needed to reach the waypoint.

7. What would happen if an obstacle was between the robot and its goal?

The robot would run into the obstacle and not reach its goal given that there is no obstacle avoidance code. The robot is simply programmed to reach set waypoints by turning towards it, then traveling directly at it.

8. (Briefly) How would you implement simple obstacle avoidance using distance sensors if they were placed around the Turtlebot?

If an obstacle is sensed we could move the robot left, right, or backwards. Then the robot can safely move toward the waypoint and readjust it's calculations.

9. Roughly how much time did you spend programming this lab?

We spent around 8 hours programming this lab.