Part 7 – Circle & Figure 8:

1. **In the following space, describe how you designed the moveCircle() function with the flexibility of adjusting movement based upon the desired diameter:**

The moveCircle() function uses a calculated turning radius, based upon the wheels, and a calculated fudge factor for the wheel speed, that allows for adjustment of how quickly the motors spin and move.
This allows for the program to be easily adjusted if it's found the diameter of the circle is too large or too small.

2. **What is the diameter and circumference of the robot wheels? How did you use these in a formula to design the circle or turn function?**

The wheel diameter is 8.5cm, and the wheel circumference is approximately 26.7 cm. This was used with a center to center distance of 21.5 cm between the wheels to calculate how far the wheels will move along the larger or inner circle (created by the movement of the wheels).

3. **How did you modify the function to work for the figure eight?**

The Figure8 function actually just calls the circle function twice, with the second one curving the opposite direction of the original function.

Part 8 – Go To Angle Behavior:

1. **In the following space, describe how you designed the GoToAngle() function with the flexibility of adjusting movement based upon the desired angle.**

GoToAngle() uses a calculated factor based on the wheel diameter and distance to move the robot about it's center point and move to a specific angle. Both wheels move approximately 6 steps for every 1 degree of robot turning.

2. **What type of accuracy/error did you have in the go-to-angle behavior**?

GoToAngle() could be made accurate for small angles (90 degrees or less), but then it would be less accurate for large angles (720 or greater). However, if it was made accurate for larger angles, then it would be less accurate for smaller angles. It was decided to keep smaller angles accurate, since this is liksely what the robot will be dealing with.

3. **How did you calculate the turn angle for the robot? Explain and show formula in memo**

angle*angleConversion = number of steps
Angle conversion was calculated based on the number of steps needed for a full rotation of the motor (800), the wheel diameter (8.5cm) and the wheel distance (21.5 cm) to find an angle conversion of roughly 6 steps for every 1 degree of turning. This leads to the following equation:

(2.22 steps per degree) * (360 degrees per wheel rotation) / (26.7 cm per wheel rotation) *(21.5*3.14159 cm per robot rotation) / (360 degrees per robot rotation)
= 5.7 steps / degree

Part 9 – Go To Goal :

1. **In the following space, describe how you designed the GoToGoal() function with the flexibility of adjusting movement based upon the desired x and y position.**

The goToGoal() function utilized the goToAngle function and our goForward function to spin to a specific position, and then go forward that distance. This allows us to break the problem down into an angle and distance problem.
The distance conversion can be calculated based on the diameter of the wheel and steps per rotation of the motor.

2. **How did you calculate the move distance given the x and y position? Explain and show formula in memo?**

A root sum of squares:

Sqrt(x^2 + y^2) = total distance.

3. **What type of accuracy/error did you have in the go-to-goal behavior?**

Trying to handle negative numbers was a challenge, until the atan2 function was used to find the correct angle.

4. **What could you do to improve the accuracy of the behaviors?**

Although the encoders provided are not as accurate as the stepper motors, if the encoders were accurate, then a feedback system could be used to go to the exact target angle.

5. **Did your team use the turn then forward approach for go-to-goal or move and turn at the same time? If so, what were the pros and cons of using your approach versus the other one?**

Our team used the turn then forward approach. The pro was consistent and predictable movement, since if the function worked for a position of 2,2 then it would likely work for a position of -2,-2.The con, ois that the movement is relatively slow, since the robot needs to turn, then move.

Part 10 – Square Path:

1. **In the following space, describe how you designed the moveSquare() function with the flexibility of adjusting movement based upon the desired diameter.**

The moveSquare() function was designed using the goToGoal position, so each edge is consistent, with only the desired edge length having to be fed into the program.