**CHECKLIST**

**Program update to robot**: In Spike Prime select channel and click run. Update all four programs including

program 0 which will reset the fork up and with any light sensor cover, it will bring the fork down.

**Program /Track 1 : (David/Mehran)**

**Start Position**: Red Home/ 3 o’clock and the right wheel should line up with the top and end of and start from the end of the top leg.

**User Intervention:** Push the right fork pin down

**End Position:** Blue Home and Push the right pin up.

**Program /Track 2 : (David/Mehran)**

**Start Position:** Red Home/1 o’clock, and the right light sensor aim at the black and white line.  . It will follow the left edge of the black line.

**User Intervention:** Push the right pin up.

**End Position:**

**Program /Track 3 : (Mehran/David)**

**Start Position:** Red Home/12 o’clock, center left wheel on second left line  and back against wall or end of the mat (South).

**User Intervention:** Place the pusher stick in the Blue Home and the robot will wait 5 seconds to attach on the back left side.

**End Position:** Blue Home.

**Program /Track 4 : (Mehran/David)**

**Start Position:** Red Home/1 o’clock, and the right light sensor aim to the black and white line.  . It will follow the left edge of the black line.

**User Intervention:** Attach \\_\_\_\_\_\_\_/ hook, fold actors

* + 1. master piece, expert piece (anna), and 2 actors
    2. 1 actor
    3. 1 actor
    4. 2 actors
    5. 1 actor

**End Position:**

**Variables and Methods**

**Variables**

* **average\_reflection**: **55** needed to follow the line. (Max Reflection (White) + Minimum Reflection (Black))
* **cm\_to\_degree**: 1 wheel rotation is 360 degrees. Calculates degree based on distance (centimeter).
  + ***cm\_to\_degree = (distance / wheel\_circumference) \* 360***
* **correction**: left or right deviation from 0 Yaw angle provided by Gyro.
* **proportional gain**: **0.5** Proportional gain also known as Kp is the gain that applies a control action proportional to the difference between the setpoint and the actual process variable. A higher Kp results in a stronger control action and faster response to changes in the process variable.
* **relative position**: Keeps track of distance traveled (used to view distance traveled).
* **rotation**: Keeps track of rotation traveled (used to view rotation traveled).
* **target\_ black**: ?? value for black 40
* **target white**: ?? value for white 95
* **wheel\_circumference**: **17.5** Circumference of the wheel.
  + ***C=2πr***

**My Blocks: (Functions/Methods)**

* **initialize()**
* **turn(***degrees to right(+) or left(-)***)**
* **move\_fork\_by\_degree(***degrees up(+) or down(-), motor\_power***)**
* **align\_to\_line (***forward(F) or reverse(R)***)**
* **drive\_to\_target \_color (***minimum\_distance, which\_light\_sensor(L/R), color (35), motor\_power***)**
* **drive\_to\_target \_relative\_position(***distance\_to\_travel, motor\_power***)**
* **drive\_to\_target \_with\_timer(***seconds\_to\_travel, motor\_power***)**
* **follow line to intersection(***which\_light\_sensor(L/R), minimum\_distance, motor\_power***)** (follows right edge of the line only)
* **follow line\_to intersection\_by \_edge(***which\_light\_sensor(L/R), which\_edge\_of\_line(L/R), minimum\_distance, motor\_power***)**
* **follow\_line\_to\_relative\_position(***which\_light\_sensor(L/R), distance\_to\_travel, motor\_power***)** (follows right edge of the line only)
* **follow\_line\_to\_relative\_position\_by\_edge(***which\_light\_sensor(L/R) ), which\_edge\_of\_line(L/R), distance\_to\_travel, motor\_power***)**
* **follow\_line\_with\_timer(***which\_light\_sensor(L/R), seconds \_to\_travel, motor\_power*) (follows right edge of the line only)

**Notes:**

left light sensor to turn left and right light sensor to turn right.