```
1  from spike import PrimeHub, LightMatrix, Button, StatusLight, ForceSensor, MotionSensor, Speaker, ColorSensor, App,
   DistanceSensor, Motor, MotorPair
2  from spike.control import wait_for_seconds, wait_until, Timer
3  from spike.operator import equal_to, greater_than_or_equal_to, less_than_or_equal_to
4  from math import trunc, fabs, copysign
5  from utime import ticks_ms
6  import gc
7
8
9  hub = PrimeHub()
10
11 hub.light_matrix.show_image('HAPPY')
12
13 driveMotors = MotorPair('B', 'A')
14
15 attachmentMotors = MotorPair('C', 'D')
16
17 leftAttachment = Motor('C')
18 rightAttachment = Motor('D')
19
20 leftDrive = Motor ('B')
21 rightDrive = Motor ('A')
22
23 leftSensor = ColorSensor('E')
24 rightSensor = ColorSensor('F')
25
26 gc.collect()
27
28 #Just driving. Set the amount in degrees, steering (0 is straight), and power
29 def drive(amount, steering, power):
30     #setup
31     hub.light_matrix.show_image('TORTOISE')
32     driveMotors.set_stop_action('brake')
33     #execution
34     driveMotors.move(amount, 'degrees', steering, power)
35     driveMotors.stop()
36
37 #Wait for a button push, with a fancy clock
38 def wait():
39     #setup
40     driveMotors.set_stop_action('brake')
41     attachmentMotors.set_stop_action('brake')
42     clockFaces = ['CLOCK12',
   'CLOCK1','CLOCK2','CLOCK3','CLOCK4','CLOCK5','CLOCK6','CLOCK7','CLOCK8','CLOCK9','CLOCK10','CLOCK11']
43     done = False
44     index = 0
45     clockSpeed = 30
46     hub.left_button.was_pressed()
47     #execution
48     driveMotors.stop()
49     attachmentMotors.stop()
50     while not done:
51         hub.light_matrix.show_image(clockFaces[trunc(index/clockSpeed)% len(clockFaces)])
52         index += 1
53         if hub.left_button.was_pressed():
54             done = True
55
```

```
def alignWithLine():
    hub.light_matrix.show_image('DUCK')
    #Find the line
    #FindTheLine setup
    done = False
    driveMotors.set_stop_action('brake')
    driveSpeed = 15
    backUp = -8
    rightSensorValue = 100
    leftSensorValue = 100
    darkEnough = 30
    isRightBlack = False
    #FindTheLine action
    while not done:
        driveMotors.start(0, driveSpeed)
        rightSensorValue = rightSensor.get_reflected_light()
        leftSensorValue = leftSensor.get_reflected_light()
        if rightSensorValue < darkEnough or leftSensorValue < darkEnough:
            done = True
            if rightSensorValue < darkEnough:
                isRightBlack = True
                hub.light_matrix.show_image('ARROW_NW')
            else:
                isRightBlack = False
                hub.light_matrix.show_image('ARROW_SW')
    driveMotors.stop()
    wait_for_seconds(1)
    #driveMotors.move(backUp, 'degrees', driveSpeed)
    #The second sensor needs to find black
    #SecondSensor setup
    done = False
    smallerDriveSpeed = ((-driveSpeed) * 1)
    #SecondSensor action
    if isRightBlack:
        while not done:
            driveMotors.start_tank(driveSpeed, trunc(smallerDriveSpeed))
            leftSensorValue = leftSensor.get_reflected_light()
            if leftSensorValue < darkEnough:
                done = True
    else:
        while not done:
            driveMotors.start_tank(trunc(smallerDriveSpeed), driveSpeed)
            rightSensorValue = rightSensor.get_reflected_light()
            if rightSensorValue < darkEnough:
                done = True
    driveMotors.stop()
    #WIGGLE time
    hub.light_matrix.show_image('PACMAN')
    #Wiggle set-up
    time = 300
    numberOfLoops = 0
    targetValue = 61
    #Wiggle action
    while numberOfLoops < time:
        numberOfLoops += 1
        Ck = 0.5
        rightSensorValue = rightSensor.get_reflected_light()
```

```
        leftSensorValue = leftSensor.get_reflected_light()
        rightDriveAmount = (rightSensorValue - targetValue) * Ck
        leftDriveAmount = (leftSensorValue - targetValue) * Ck
        driveMotors.start_tank(trunc(leftDriveAmount), trunc(rightDriveAmount))
    driveMotors.stop()


#Turns right a set amount of degrees, not TO a specific degree
#Note: Turning right goes from 0 to 179, then -179 to 0 in degrees
def turnRightForward(endAngle, power):
    #setup
    hub.motion_sensor.reset_yaw_angle()
    hub.light_matrix.show_image('ARROW_N')
    driveMotors.set_stop_action('brake')
    #execution
    rightDrive.stop()
    #Negative powers counter clockwise
    leftDrive.start(-power)
    wait_until(hub.motion_sensor.get_yaw_angle, greater_than_or_equal_to, endAngle)
    leftDrive.stop()

#Turns left a set amount of degrees, not TO a specific degree
#Note: Turning left goes from 179 to 0, then 0 to -179 in degrees
def turnLeftForward(endAngle, power):
    #setup
    hub.motion_sensor.reset_yaw_angle()
    hub.light_matrix.show_image('ARROW_N')
    driveMotors.set_stop_action('brake')
    #execution
    leftDrive.stop()
    #Negative powers counter clockwise
    rightDrive.start(power)
    wait_until(hub.motion_sensor.get_yaw_angle, less_than_or_equal_to, -endAngle)
    rightDrive.stop()

#Turns right a set amount of degrees, not TO a specific degree, but it's backwards
def turnRightBack(endAngle, power):
    #setup
    hub.motion_sensor.reset_yaw_angle()
    #wait_for_seconds(0.7)
    hub.light_matrix.show_image('ARROW_N')
    driveMotors.set_stop_action('brake')
    #execution
    leftDrive.stop()
    #Negative powers counter clockwise
    rightDrive.start(-power)
    wait_until(hub.motion_sensor.get_yaw_angle, greater_than_or_equal_to, endAngle)
    rightDrive.stop()

#Turns left a set amount of degrees, not TO a specific degree, but it's backwards
def turnLeftBack(endAngle, power):
    #setup
    hub.motion_sensor.reset_yaw_angle()
    #wait_for_seconds(0.7)
    hub.light_matrix.show_image('ARROW_N')
    driveMotors.set_stop_action('brake')
    #execution
```

```
    rightDrive.stop()
    #Negative powers counter clockwise
    leftDrive.start(power)
    wait_until(hub.motion_sensor.get_yaw_angle, less_than_or_equal_to, -endAngle)
    leftDrive.stop()


#Turns right a set amount of degrees using tank, not TO a specific degree
#Note: Turning right goes from 0 to 179, then -179 to 0 in degrees
def turnRightTank(endAngle, power):
    #setup
    hub.motion_sensor.reset_yaw_angle()
    hub.light_matrix.show_image('DUCK')
    driveMotors.set_stop_action('brake')
    steering = 100
    #execution
    currentAngle = hub.motion_sensor.get_yaw_angle()
    while currentAngle <= endAngle:
        currentAngle = hub.motion_sensor.get_yaw_angle()
        driveMotors.start_tank(power, -power)
    driveMotors.stop()

#Turns left a set amount of degrees using tank, not TO a specific degree
#Note: Turning left goes from 179 to 0, then 0 to -179 in degrees
def turnLeftTank(endAngle, power):
    #setup
    hub.motion_sensor.reset_yaw_angle()
    hub.light_matrix.show_image('DUCK')
    driveMotors.set_stop_action('brake')
    steering = -100
    count = 0
    #execution
    currentAngle = hub.motion_sensor.get_yaw_angle()
    while currentAngle >= -endAngle:
        currentAngle = hub.motion_sensor.get_yaw_angle()
        driveMotors.start_tank(-power, power)
    driveMotors.stop()

#Finds black on the board
def findBlackTurning(steering, power, lightIntensity, lightSensor):
    #setup
    hub.light_matrix.show_image('SKULL')
    driveMotors.set_stop_action('brake')
    #steering=0
    #execution
    driveMotors.start(steering, power)
    wait_until(lightSensor.get_reflected_light, less_than_or_equal_to, lightIntensity)
    driveMotors.stop()

def findBlack(power, lightIntensity, lightSensor):
    #setup
    hub.light_matrix.show_image('SKULL')
    driveMotors.set_stop_action('brake')
    steering=0
    #execution
    driveMotors.start(steering, power)
    wait_until(lightSensor.get_reflected_light, less_than_or_equal_to, lightIntensity)
```

```
227     driveMotors.stop()
228
229
230
231 def driveStraightForSeconds(power, timeInSeconds):
232     #-177 is limit
233     #setup
234     goal = 0
235     hub.light_matrix.show_image('GHOST')
236     hub.motion_sensor.reset_yaw_angle()
237     done = False
238     Kp = 3
239     timer = Timer()
240     timer.reset()
241     yaw = hub.motion_sensor.get_yaw_angle()
242     #execution
243     while not done:
244         yaw = hub.motion_sensor.get_yaw_angle()
245         error = goal-yaw
246         P=Kp*error
247         driveMotors.start(P, power)
248         if timeInSeconds < timer.now():
249             driveMotors.stop()
250             done=True
251
252 def timeFollower1(lightSensor, driveTime, speed, direction):
253     #Good numbers below
254     #Kp for proportional only is 2.2
255     #Kp = 1.1, ki = 0.02, kd = 20, iClamp = 1000, ifClamp = 15, oClamp = 65
256     #8/27 good numbers: speed=25 steering=0, goal=68, kp=1.3, ki=0.1, kd=10, iClamp=99, ifClamp=30, oClamp=65, dClamp=80
257
258     #PREPARATIONS!!
259     #Speed, steering and goal
260     #speed = 20 #15 #20 #25 #20 #15 #10 #35 #40 #45 #40 #35 #30 #25 #30 #25 #30 #40 #25
261     steering = 0
262     goal = 68
263     percent = goal
264     startTime = 0
265     endTime = 0
266
267     #Errors
268     error = 0
269     totalError = 0
270     lastError = 0
271
272     #Defining
273     pFix = 0
274     iFix = 0
275     dFix = 0
276     overall = 0
277
278     #The Ks
279     #Kp only is 2
280     #kp = 1.9
281     #kp = 0.8, kd = 0.9
282     Kp = 1.3 #0.9 #1 #0.9 #1.4 #1 #0.9 #0.8 #0.7 #1 #0.9 #1.8 #1.9 #1.75 #1.5 #1.4 #2 #2.1 #2.2 #2 #1.3 #1.1 #0.9 #0.8 #1.2 #1.1
#1.3
283     Ki = 0.1 #0.025 #0 #0.025 #0 #0.025 #0.05 #0.1 #0.05 #0.1 #0.1 #0.2 #0.02 #0.002
```

```
    Kd = 10 #0.5 #80 #100 #1 #0.5 #0.25 #0.5 #1 #2 #4 #5 #10 #5 #1.1 #1 #0.9 #0.8 #0.75 #0.5 #1 #2 #5 #15 #10 #10 #15 #20 #10 #25 #20 #10 #15

    #Clamping
    iClamp = 99 #1000 #99 #1000
    ifClamp = 30 #15 #30 #15 #30 #15
    oClamp = 65
    dClamp = 80 #40 #80 #40 #99

    #Time Stuff
    sampleTime = 23 #14
    #Moved to ms, are using ticks_ms
    #Old time values are in seconds 0.01 #0.004 #0.005 #0.004 #0.003 #0.004 #0.005 #0.002 #0.003 #0.006 #0.001 #0.005 #0.01 #0.001
    timer = Timer()
    driveEndTime = 0

    if lightSensor == leftSensor:
        if direction == 'inside':
            directionMult = -1
        elif direction == 'outside':
            directionMult = 1
    elif lightSensor == rightSensor:
        if direction == 'inside':
            directionMult = 1
        if direction == 'outside':
            directionMult = -1

    #EXECUTE!!
    #Image
    hub.light_matrix.show_image('GIRAFFE')

    #Start the motors!
    driveMotors.start(steering, speed)

    #Print so we know when it starts
    #print('**************************************************************')

    timer.reset()

    #Loop!

    done = False
    #driveTime is being multiplied by 1000 because driveTime is seconds and ticks_ms is miliseconds
    driveEndTime = ticks_ms() + driveTime * 1000
    while not done:
        startTime = ticks_ms()
        #Recording light reflectiveness
        percent = lightSensor.get_reflected_light()

        #Proportianal
        # there is no need for if statements because if percent is more then the robot will go
        # left and if percent is less then itll go right
        error = goal - percent
        pFix = Kp * (error)

        #Integral
        totalError = totalError + error
```

```
340         if fabs(totalError) > iClamp:
341             totalError = copysign(iClamp, totalError)
342         iFix = totalError * Ki
343         if fabs(iFix) > ifClamp:
344             iFix = copysign(ifClamp, totalError)
345 
346         #Derivative
347         dFix = (error - lastError) * Kd
348         if fabs(dFix) > dClamp:
349             dFix = copysign(dClamp, dFix)
350         lastError = error
351 
352         #Overall
353         overall = pFix + iFix + dFix
354         if fabs(overall) > oClamp:
355             overall = copysign(oClamp, overall)
356 
357         #Start
358         driveMotors.start(trunc(overall * directionMult), speed)
359         endTime = ticks_ms()
360         #print(startTime - endTime, ', ')
361         #wait_for_seconds(time)
362         #Printing for troubleshooting
363         #print('totalError:', totalError, ' iFix:', iFix, ' dFix:', dFix, ' pFix:', pFix, ' overall:', overall)
364 
365 
366         if ticks_ms() > driveEndTime:
367             hub.light_matrix.show_image('DIAMOND')
368             done = True
369 
370         while ticks_ms() < startTime + sampleTime:
371             pass
372 
373 
374 
375 def followLineForTime(lightSensor, driveTime, direction, chooseStop):
376     timeFollower1(lightSensor, 0.5, 20, direction) #1 2
377     timeFollower1(lightSensor, driveTime - 0.5, 30, direction) #32 #30 #35
378     driveMotors.set_stop_action(chooseStop)
379     driveMotors.stop()
380 
381 def lineFollower1(lightSensor, driveTime, speed, direction):
382     #Good numbers below
383     #Kp for proportional only is 2.2
384     #Kp = 1.1, ki = 0.02, kd = 20, iClamp = 1000, ifClamp = 15, oClamp = 65
385     #8/27 good numbers: speed=25 steering=0, goal=68, kp=1.3, ki=0.1, kd=10, iClamp=99, ifClamp=30, oClamp=65, dClamp=80
386 
387     #PREPARATIONS!!
388     #Speed, steering and goal
389     #speed = 20 #15 #20 #25 #20 #15 #10 #35 #40 #45 #40 #35 #30 #25 #30 #25 #30 #40 #25
390     steering = 0
391     goal = 68
392     percent = goal
393 
394     #Errors
395     error = 0
396     totalError = 0
```

```
    lastError = 0

    #Defining
    pFix = 0
    iFix = 0
    dFix = 0
    overall = 0

    #The Ks
    #Kp only is 2
    #kp = 1.9
    #kp = 0.8, kd = 0.9
    Kp = 1.3 #0.9 #1 #0.9 #1.4 #1 #0.9 #0.8 #0.7 #1 #0.9 #1.8 #1.9 #1.75 #1.5 #1.4 #2 #2.1 #2.2 #2 #1.3 #1.1 #0.9 #0.8 #1.2 #1.1 #1.3
    Ki = 0.1 #0.025 #0 #0.025 #0 #0.025 #0.05 #0.1 #0.05 #0.1 #0.1 #0.2 #0.02 #0.002
    Kd = 10 #0.5 #80 #100 #1 #0.5 #0.25 #0.5 #1 #2 #4 #5 #10 #5 #1.1 #1 #0.9 #0.8 #0.75 #0.5 #1 #2 #5 #15 #10 #10 #15 #20 #10 #25 #20 #10 #15

    #Clamping
    iClamp = 99 #1000 #99 #1000
    ifClamp = 30 #15 #30 #15 #30 #15
    oClamp = 65
    dClamp = 80 #40 #80 #40 #99

    #Time Stuff
    sampleTime = 23 #20 #36 #25 #50 #40 #30 #14
    #Moved to ms, are using ticks_ms
    #Old time values 0.004 #0.005 #0.004 #0.003 #0.004 #0.005 #0.002 #0.003 #0.006 #0.001 #0.005 #0.01 #0.001
    timer = Timer()

    #Inside and Outside
    if lightSensor == leftSensor:
        if direction == 'inside':
            directionMult = -1
        elif direction == 'outside':
            directionMult = 1
    elif lightSensor == rightSensor:
        if direction == 'inside':
            directionMult = 1
        if direction == 'outside':
            directionMult = -1

    #Light sensor stuff
    if lightSensor == leftSensor:
        light_stop_sensor = rightSensor
    elif lightSensor == rightSensor:
        light_stop_sensor = leftSensor

    #EXECUTE!!
    #Image
    hub.light_matrix.show_image('YES')

    #Start the motors!
    driveMotors.start(steering, speed)

    #Timer Reset
    timer.reset()

```

```
    #Loop!
    done = False
    #driveTime is being multiplied by 1000 because driveTime is seconds and ticks_ms is miliseconds
    driveEndTime = ticks_ms() + driveTime * 1000
    while not done:
        startTime = ticks_ms()

        #Recording light reflectiveness
        percent = lightSensor.get_reflected_light()
        percentStop = light_stop_sensor.get_reflected_light()

        #Proportianal
        # there is no need for if statements because if percent is more then the robot will go
        # left and if percent is less then itll go right
        error = goal - percent
        pFix = Kp * (error)

        #Integral
        totalError = totalError + error
        if fabs(totalError) > iClamp:
            totalError = copysign(iClamp, totalError)
        iFix = totalError * Ki
        if fabs(iFix) > ifClamp:
            iFix = copysign(ifClamp, totalError)

        #Derivative
        dFix = (error - lastError) * Kd
        if fabs(dFix) > dClamp:
            dFix = copysign(dClamp, dFix)
        lastError = error

        #Overall
        overall = pFix + iFix + dFix
        if fabs(overall) > oClamp:
            overall = copysign(oClamp, overall)

        #Start
        driveMotors.start(trunc(overall * directionMult), speed)
        endTime = ticks_ms()
        #wait_for_seconds(time)

        #Stopping for other sensor seeing black
        if percentStop <= 30:
            hub.light_matrix.show_image('ARROW_W')
            done = True

        #Waiting for time then ending the loop
        if ticks_ms() > driveEndTime:
            hub.light_matrix.show_image('DIAMOND')
            done = True

        while ticks_ms() < startTime + sampleTime:
            pass


#Defining Line Follower
def followLineUntilNextLine(lightSensor, driveTime, direction, chooseStop):
```

```
    lineFollower1(lightSensor, 0.5, 20, direction) #1 2
    lineFollower1(lightSensor, driveTime - 0.5, 30, direction) #32 #35
    driveMotors.set_stop_action(chooseStop)
    driveMotors.stop()


def lineFollowTesterer():
    drive(120, 0, 50) #Forward drive before we find the line
    #driveMotors.move_tank(147, 'degrees', 30, 0) #90 degree turn right for line
    turnRightForward(86, 30) #Turn towards the line before we find it
    findBlack(20, 25, rightSensor) #Drive froward to find the line and stops
    followLineForTime(rightSensor, 9.375, 'outside', 'brake') #Follows the line until after the line in the wheel flip mission
    #driveMotors.stop()
    #wait()
    followLineUntilNextLine(rightSensor, 3,'outside', 'brake') #Follows the line until detects the line near the treadmill

def moveRake(upOrDown, rakeTime, rakePower):
    if upOrDown == 'up':
        rakePower = -rakePower
    elif upOrDown == 'down':
        rakePower = rakePower
    else:
        print('Incorrect phrasing, make sure it is either up or down')
    rightAttachment.start(rakePower)
    wait_for_seconds(rakeTime)
    rightAttachment.stop()

def moveRakeForDegrees(rakeDegrees, rakePower):
    rightAttachment.run_for_degrees(-rakeDegrees, rakePower)

def moveLeftForDegrees(leftDegrees, leftPower):
    leftAttachment.run_for_degrees(-leftDegrees, leftPower)

def moveLeft(leftSeconds, leftPower):
    leftAttachment.run_for_seconds(leftSeconds, -leftPower)

#
#|¯¯|    (A function goes in a program)    /\
#||    Programs below|Functions above    /\
#||    Programs below|Functions above/    \
#||    Programs below|Functions above/    \
#||    Programs below|Functions above¯¯||¯¯
# __||__Programs below|Functions above    ||
# \    /Programs below|Functions above    ||
#\    /Programs below|Functions above    ||
#\/    Programs below|Functions above    ||
#    \/    (A program is made up of functions)|__|
#


#robot run 2
def rakeForklift():
    drive(400, 0, 50)
    moveRake('up', 1, 20)
    drive(-200, 0, 50)

#rakeForklift()
```

```
#moveRake('down', 0.6)




def boxDropOff():
    drive(-200, 0, 30)
    moveRakeForDegrees(50, 50)
    drive(200, 0, 30)
    moveRakeForDegrees(100, 50)
    drive(50, 0, 30)
    moveRakeForDegrees(210, 50)

#
# ░░░░░░░░░░░░░░░░░░░░░░░░░░
# ░░░░         ░░░░░░░░░░
# ░░░░         ░░░░░░░░░░
# ░░░░░░░░░░   ░░░░░░░░░░
# ░░░░░░░░░░   ░░░░░░░░░░
# ░░░░░░░░░░   ░░░░░░░░░░
# ░░░░░░░░░░   ░░░░░░░░░░
# ░░░░░░░░░░   ░░░░░░░░░░
# ░░░░░░░░░░   ░░░░░░░░░░
# ░░░░            ░░░░
# ░░░░            ░░░░
# ░░░░░░░░░░░░░░░░░░░░░░░░░░
#

def oneDelivery():
    #drivng forward from launch area
    drive(400,0,45) #360 370 380 400 375
    #turns right
    turnRightForward(48,30) #55 #45
    wait_for_seconds(0.2)
    #drives forward to cargo connect area
    #wait()
    drive(770,0,45) #700 580 800 750 650 760
    #turns toward the cargo connect area
    turnRightForward(20,30) #24 #26 #18 20 #16
    #drives into the cargo connect area
    #wait()
    findBlack(45, 40, leftSensor)
    drive(475, 0, 45) #750 800 700 800 850 900 1200 775 575
    #lines up to drop off inovative attachment
    turnRightForward(53,30) #53 60 63 43 85
    #drives into cargo connect area
    drive(40, 0, 40) #50 100 130 100
    #releases the innovative attachment into the cargo connect area
    moveRakeForDegrees(440,55)#140
    #backs out of cargo connect area
    drive(162, 0, -40) #142 100 165 185 200
    #turnRightTank(15,15)
    #turnLeftTank(15,15)
    #wait_for_seconds(0.2)#wait()
    #turns left to go foward to the train tracks
```

```
    turnLeftBack(21, 30) #23 25 23 15 16 35 85
    #drives foward to train tracks
    drive(150,0,45)
    wait_for_seconds(0.2) #wait()
    findBlack(25,30,rightSensor)
    #drive(225, 0, 45) # we used this before we used find black ^^#500-before turn 550 450 400 350
    #turns left to aligns itself a little more
    #wait()
    turnLeftForward(20, 20) #14 16 10
    wait_for_seconds(0.2)
    #drives a little more foward toward the train tracks
    drive(130, 0, 45) #125 225 35 215 208 230 200 300 175
    #turns toward the sorting center
    turnRightForward(79, 30) #80 78 73 76 83 80 70 53 55 85 70 75
    #wait()
    wait_for_seconds(0.2) #wait()
    #rake goes down so can get blocks
    moveRake('down', 0.8, 55) #0.5
    wait_for_seconds(0.5)
    #drives into sorting center and gets blocks
    driveStraightForSeconds(25, 2)
    #has attachments lift down the train tracks and the package gets delivered
    leftAttachment.run_for_seconds(1,-100)
    moveRakeForDegrees(180,100)
    wait_for_seconds(0.5) #0.2
    #rake pickes up blocks
    moveRake('up', 1, 100)
    wait_for_seconds(0.2) #wait()
    #the ramp and the hook goes back up
    leftAttachment.run_for_seconds(1, 45)
    #backs out of the sorting center
    drive(350,0,-45)
    #turns right to go back home
    turnRightBack(70, 30) #7072 70 84 74 72 71
    wait_for_seconds(0.2) #wait()
    #drives back home
    #wait()
    drive(1700, -5, 70) #700 -5
    #has the robot align to get back into home straight
    turnRightForward(12, 45) #10 30
    #turnLeftForward(48, 30)
    #turnRightForward(48, 30)
    #gets into home fully
    drive(1000, 6, 60) #200 400 600
    #turnRightForward(70, 40)
    wait_for_seconds(0.2) #wait()
    moveRake('down',1,80)

#leftAttachment.run_for_seconds(0.5, -35)
#wait()
#leftAttachment.run_for_seconds(2.7, -97)
#oneDelivery()

#
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
# ▒▒▒▒▒▒▒                  ▒▒▒▒▒▒▒
# ▒▒▒▒▒▒▒                  ▒▒▒▒▒▒▒
```

```
# ░░░░░░░░░░░░░░░░░░░░░░░░░░░  ░░░░░░
# ░░░░░░░░░░░░░░░░░░░░░░░░░░░  ░░░░░░
# ░░░░░░          ░░░░░░
# ░░░░░░          ░░░░░░
# ░░░░░░   ░░░░░░░░░░░░░░░░░░░░░░░░░░
# ░░░░░░   ░░░░░░░░░░░░░░░░░░░░░░░░░░
# ░░░░░░          ░░░░░░
# ░░░░░░          ░░░░░░
# ░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░
#

def twoPlace():
    moveRake('down', 1, 100)
    wait()
    leftAttachment.start(25)
    drive(330, 0, 45) #350 300
    #turn to avoid the bridge
    turnRightForward(45, 30)
    leftAttachment.stop()
    drive(750, 0, 45) #675 655 600 650 700 750 800
    turnRightForward(26, 30) #27 24 23 22
    #drive(235, 0, 45) #535 530 700 800 850 800, 850 750 700
    #look for line in frount of bridge
    findBlack(45, 40, leftSensor)
    #drive toward cargo deck
    drive(380, 0, 45)#100 300 320 350 370 365 375 385
    turnLeftTank(78, 15)#80, 78, 79
    wait_for_seconds(0.5)
    #turnLeftForward(80, 30) #73 71 69 65
    #Load cargo onto balancing deck
    driveStraightForSeconds(35, 1.25)
    driveStraightForSeconds(15, .25)
    moveRakeForDegrees(360, 50)
    drive(260, 0, -45)#250 270
    #move the hook down
    leftAttachment.start(-100)
    turnRightForward(82, 30) #70 75 78 75 70 80
    drive(30, 0, -25) #150 55 50 40 35
    leftAttachment.stop()
    #drive forward to unload cargo
    drive(200, 0, 40) #280 300
    moveRakeForDegrees(-360, 50)
    drive(200, 0, -45)
    #turn away from cargo deck
    turnRightForward(6, 30) #7 8 5
    #lift up hook
    leftAttachment.start(100)
    wait_for_seconds(0.5)
    leftAttachment.stop()
    #drive toward train
    drive(520, 0, 45) #750 500 480 470 490 600 530
    #big boi turn
    turnRightTank(65, 30) #80 82 85 83 88 86 84 67 70 75 73 68
    #back into wall
    driveMotors.start(0, -50)
    wait_for_seconds(1.5)#1 2
```

```
    driveMotors.stop()
    #drive(590, 0, -50) #600 570
    drive(100, 0, 45) #200 150 170 180 150
    turnLeftTank(35, 15)#20 30
    #drop off blue block
    moveRakeForDegrees(150, 90) #90 100
    drive(100, 0, 45) #100 50
    #turn and release package
    turnRightTank(100, 15) #80 40 100
    #back up into helicopter
    drive(300, 0, -45) #150 100
    #turnRightTank(15, 15)
    #turnLeftTank(15, 15)
    drive(100, 0, 45)
    #turn so that the hook has room to go down
    turnLeftTank(45, 15) #80 40 53 68 58 60 40
    #back up toward helicopter
    drive(45, 0, -50) #200 100 75 35
    wait()
    #push da train
    leftAttachment.start(-100)
    driveStraightForSeconds(30, 3)
    leftAttachment.stop()
    drive(200, 0, -50)



#twoPlace()
#or to not to place. That is the question



#
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
# ▒▒▒▒▒▒                  ▒▒▒▒▒▒
# ▒▒▒▒▒▒                  ▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒    ▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒    ▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒            ▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒            ▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒    ▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒    ▒▒▒▒▒▒
# ▒▒▒▒▒▒                  ▒▒▒▒▒▒
# ▒▒▒▒▒▒                  ▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
#
def threeMusketeers():
    #moveRake('up', 1, 55)
    leftAttachment.start(6)
    rightAttachment.start(-50)
    driveStraightForSeconds(62, 2.73)#hits wall 2.5 3.5, 55
    leftAttachment.stop()
    rightAttachment.stop()
    turnRightTank(30,25)#flips switch engine over
    turnLeftTank(30,25)#turns to wall
```

```
    wait_for_seconds(0.5) #1
    driveStraightForSeconds(25,0.5)#run into wall
    drive(95,0,-45) #85 65 95 75 #55 #backing away from wall
    turnRightBack(35,42) #25 #45 #15 #42#turn so back is facing home
    drive(371,0,-45) #new idea 255 285 #275 #270 #260 #240#back up to handle to cargo ship
    #wait_for_seconds(0.1)
    #Putting light saber at the right place
    leftAttachment.run_for_degrees(81, -60) #68 76 81
    #wait()
    wait_for_seconds(0.32)
    #wait()
    #driving to air lift
    drive(100, 0, 45)
    #wait()
    wait_for_seconds(0.1)
    #grabs handle and puts it down
    leftAttachment.start(-100)#-60 -80
    wait_for_seconds(0.2) #0.1 0.5
    turnRightBack(45,40)#turns toward the cargo ship 45
    #wait()
    leftAttachment.run_for_seconds(1,50)#puts the attachment up
    turnLeftBack(45,45)#turn towards the cargo
    #wait()
    drive(222,0,25) # 200 150
    moveRake('down',0.9,100) #55 1.5
    drive(162,0,-50) #142 122 118
    moveRake('up',0.85,100) #55 1.5
    drive(653,0,-70) #553 245 250 275 300 400 500 -50

#rightAttachment.run_for_degrees(54, -60)




threeMusketeers()

#
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
# ▒▒▒▒▒▒    ▒▒▒▒▒▒▒▒▒▒▒▒    ▒▒▒▒▒▒
# ▒▒▒▒▒▒    ▒▒▒▒▒▒▒▒▒▒▒▒    ▒▒▒▒▒▒
# ▒▒▒▒▒▒    ▒▒▒▒▒▒▒▒▒▒▒▒    ▒▒▒▒▒▒
# ▒▒▒▒▒▒    ▒▒▒▒▒▒▒▒▒▒▒▒    ▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒              ▒▒▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒              ▒▒▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒    ▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒    ▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒    ▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒    ▒▒▒▒▒▒
# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
#

def acc_avoid():
    #driveStraightForSeconds(-40,2)
    driveMotors.start(0, -50)
```

```
    wait_for_seconds(1.5)#1 2
    driveMotors.stop()
    drive(50, 0, 20)
    rightAttachment.start(-100)
    turnRightTank(77, 15)
    rightAttachment.stop()
    moveRakeForDegrees(-100,100)
    driveStraightForSeconds(10,2)


def transportationJourney():
    leftAttachment.run_for_seconds(1.5,-100)#1
    leftAttachment.start(100)
    #leftAttachment.run_for_seconds(1.25,100)#3
    wait_for_seconds(1)
    leftAttachment.stop
    drive(440, 0, 50)#390
    turnLeftForward(3.93,15) #3.35 2.8 2.4 2
    leftAttachment.run_for_seconds(1,-100)
    #leftAttachment.run_for_seconds(3,100)
    leftAttachment.start(100)
    wait_for_seconds(1)
    leftAttachment.stop
    drive(340, 0, 50)#375 275 250
    turnRightTank(109, 15)

#transportationJourney()

def truck():
    drive(458, 0, 50) #575 390 455 460
    turnRightForward(80, 20) #82 35
    drive(400, 0, 30)
    moveRake('up', 0.5, 50)
    drive(100, 0, -50) #500 100
    drive(100,0,-40)
    turnLeftBack(17, 20) #40
    drive(869, 0, 30) #669 before took out transpo jounrey 169
    turnRightTank(115,15)

def rakeSetup():
    wait()
    moveRake('down',0.5,100)
    wait()

def fourKablam():
    rakeSetup()
    truck()
    acc_avoid()

#fourKablam()

def masterProgram():
    #setup
    runNumber = ['4','3','2','1']
    index = 3
    hub.left_button.was_pressed()
    hub.right_button.was_pressed()
```

```
902     #execution
903     while True:
904         hub.light_matrix.write(runNumber[index%4])
905         if hub.right_button.was_pressed():
906             index += 1
907         if hub.left_button.was_pressed():
908             if index%4 == 3:
909                 oneDelivery()
910                 index += 3
911             elif index%4 == 2:
912                 twoPlace()
913                 index += 3
914             elif index%4 == 1:
915                 threeMusketeers()
916                 index += 3
917             elif index%4 == 0:
918                 fourKablam()
919                 index += 3
920             hub.left_button.was_pressed
921         wait_for_seconds(0.1)
922 
923 #masterProgram()
```