❖ Paper Control Robot

```
#-------------Setup----------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
#Note: this program simply reacts to the messages sent by the pen control Edison.
#Program the Edison controlling the paper with this program.
#This program should not need any modification to allow the printer to draw different shapes.

#Press 'play' on this robot BEFORE you press 'play' on the pen-control Edison.


#event handler for event 'printer_receive' - constantly monitoring for the event
Ed.RegisterEventHandler(Ed.EVENT_IR_DATA, "printer_receive")
#forever
while True:
    pass

#definition for 'printer_receive()' function, reads the message received from the pen control Edison, if any
def printer_receive():
    message = Ed.ReadIRData()

    #check for direction flags (set by pen control Edison)
    if message>64:
        #"Drive forwards" flag found. Remove the flag from the message
        message = message-64
        #drive the requested distance
        Ed.Drive(Ed.FORWARD, 1, message)
        #send a message to indicate the drive is complete
        Ed.SendIRData(5)
    elif message>32:
        #"Drive backwards" flag found. Remove the flag from the message
        message = message-32
        #drive the requested distance
        Ed.Drive(Ed.BACKWARD, 1, message)
        #send a message to indicate the drive is complete
        Ed.SendIRData(5)

```

❖ Pen Control - 1

```
#-------------Setup----------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
#Program the Edison controlling the pen with this program.
#This program will create a rectangle. You can also modify this program to create other shapes.

#NOTE: due to the printer gearing, using 15cm as the input parameter in the drive function results in a ~4.5cm movement of the pen.

#For the rectangle program, start with the pen as close to the pen-control robot as possible.

#Push 'play' on this robot AFTER you push 'play' on the paper-control Edison.


#event handler for event 'message_receive' - constantly monitoring for the event
Ed.RegisterEventHandler(Ed.EVENT_IR_DATA, "message_receive" )
messageReceivedFlag = 1

#create a rectangle progam
drawLineLeft(15)
drawLineForward(4)
drawLineRight(15)
drawLineBackward(4)


#Pen-control Edison base functions


#definition of 'drawLineLeft(numCM)' function, draw a line moving away from Edison
def drawLineLeft(numCM):
    #constrain input value
    if numCM > 15:
        numCM = 15
    #move pen
    Ed.Drive(Ed.FORWARD, 2, numCM)

#definition of 'drawLineRight(numCM)' function, draw a line moving towards Edison
def drawLineRight(numCM):
    #constrain input value
    if numCM > 15:
        numCM = 15
    #move pen
    Ed.Drive(Ed.BACKWARD, 2, numCM)

#definition of 'drawLineForward(numCM)' function, move the paper to draw a line forwards on the paper
def drawLineForward(numCM):
    #constrain input value
    if numCM > 15:
        numCM = 15

    #set send message with "drive forwards" flag
    sendValue = 64
    #Add distance to drive to the message, using a 'bitwise OR'
    sendValue = sendValue|numCM

    #send message to move paper to the paper-controlling Edison
    Ed.SendIRData(sendValue)
    #wait for the paper-controlling Edison to send a message to indicate it has stopped moving
    dataBack = 255;
    while dataBack != 5:
        dataBack= waitForMessage()

#definition of 'drawLineBackward(numCM)' function, move the paper to draw a line backwards on the paper
def drawLineBackward(numCM):
    #constrain input value
    if numCM > 15:
        numCM = 15

    #set send message with "drive backwards" flag
    sendValue = 32
    #add distance to drive to the message, using a 'bitwise OR'
    sendValue = sendValue|numCM

    #send message to move paper
    Ed.SendIRData(sendValue)
    #wait for the paper-controlling Edison to send a message to indicate it has stopped moving
    dataBack = 255;
```

❖ Pen Control - 2

```
85      while dataBack != 5:
86          dataBack= waitForMessage()
87
88  #definition of 'waitForMessage()' function, to wait for a message to be seen before returning the value of the sent message
89  def waitForMessage():
90      global messageReceivedFlag
91      while messageReceivedFlag==0:
92          pass
93      messageReceivedFlag=0
94      return Ed.ReadIRData()
95
96  #definition of 'message_receive()' function, sets the message received flag when a new message has been received
97  def message_receive():
98      global messageReceivedFlag
99      messageReceivedFlag = 1
```