```
"""

4.11.3. Exercise

Now write a more general function called parallelogram that draws a quadrilateral with parallel sic

"""

import turtle
import sys

class Window():
    def __init__(self):

        #create the canvas and turtle
        self.t = turtle.Turtle()

        #This dictionary contains the action step and it's inverse function
        self.actions = {
            "left" : self.t.right,
            "right" : self.t.left,
            "forward" : self.t.back,
            "backward" : self.t.forward
        }

        #track all of the steps taken in a dictionary.
        self.steps = []

    def parallelogram(self, l1=150, l2=100, angle=60, shift=0):
        """
        Takes in a width and height and draws a rectangle based on those dims.
        """
        #take the steps required to make a rectangle
        self.t.forward(l1)
        self.steps.append(("forward", l1))

        self.t.left(angle)
        self.steps.append(("left", angle))

        self.t.forward(l2)
        self.steps.append(("forward", l2))

        self.t.left(180-angle)
        self.steps.append(("left", 180-angle))

        self.t.forward(l1)
        self.steps.append(("forward", l1))

        self.t.left(angle)
        self.steps.append(("left", angle))

        self.t.forward(l2)
        self.steps.append(("forward", l2))

        if shift:
            self.t.left(shift)
            self.steps.append(("left", shift))
    def is_drawn(self):
        #Track whether a rectangle has been drawn, this will evaluate as true if self.steps > 0
        return len(self.steps)

    def undo(self):
        """
        If the rhombus is drawn, iterate through the drawing steps and complete the inverse.
        """
        if self.is_drawn():
            self.steps.reverse()
            self.t.pencolor("white")
            for step in self.steps:
                #we first need to invert the direction we are traveling in
                action = self.actions[step[0]]
                action(int(step[1]))
            self.steps = []
            self.t.pencolor("black")
        else:
            print("Nothing to undo.")


if __name__ == '__main__':

    #Instantiate the window
    w = Window()

    # Continuously prompt the user for rectangle inputs until the enter "exit"
    while 1:
        vars = input("Enter side length 1, side length 2, and interior angle separated by a space,

        try:
            if len(vars.split(" ")) > 3:

                #allowing for special 'iterations' and 'shift' keywords that will repeat the draw s
                #this makes some cool patterns
                l1, l2, angle, iterations, shift = vars.split(" ")
                l1, l2, angle, iterations, shift = int(l1), int(l2), int(angle), int(iterations), i

                for i in range(iterations):
                    w.parallelogram(l1, l2, angle, shift)

            elif len(vars.split(" ")) == 3:
                l1, l2, angle = vars.split(" ")
                l1, l2, angle = int(l1), int(l2), int(angle)
                w.parallelogram(l1, l2, angle)

            elif len(vars.split(" ")) == 1:
                if vars == "undo":
                    w.undo()
                elif vars == "exit":
                    break
        except:
            print("There was an error in the width and height entry, try again.")

    #this keeps the canvas open
    turtle.done()
```

```
sys.exit()
```