HW5

Reading: Chapter 10

Problems:

Use the usual naming scheme. Make sure you include the name(s) of any collaborators and run the doctest on your solutions.

1. Pascal's triangle is an infinite two-dimensional pattern of numbers whose first six lines are:

usual formatting:

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5   10  10  5   1
```

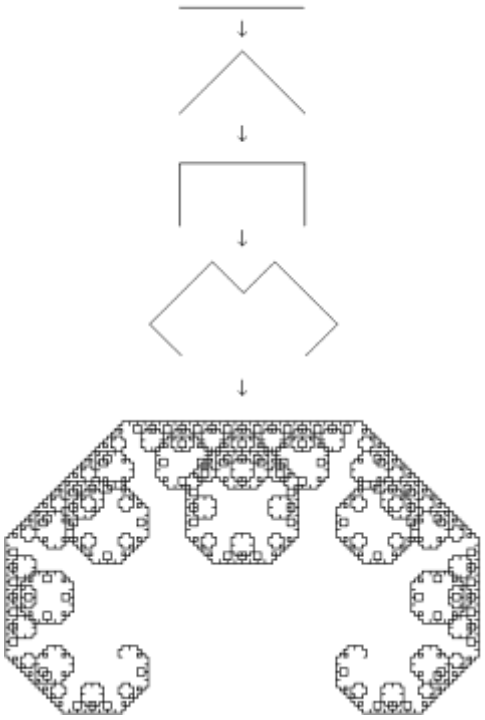
alternate formatting (may be helpful for code):

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5   10  10  5   1
```

The first line, line 0, contains just 1. All other lines start and end with a 1 as well. The other numbers in those lines are obtained using the following rule: the number at position i is the sum of the numbers in position i-1 and i in the previous line. Implement a recursive function `pascalLine` takes as a parameter a non-negative integer n and **returns** a list containing the sequence of numbers appearing in the nth line of Pascal's Triangle. The function will have one loop in it to construct the list corresponding to the line returned, but the rest of the work should be done recursively. The following shows some sample runs of the function:

```
>>> pascalLine(0)
[1]
>>> pascalLine(1)
[1, 1]
>>> pascalLine(2)
[1, 2, 1]
>>> pascalLine(3)
[1, 3, 3, 1]
>>> pascalLine(4)
[1, 4, 6, 4, 1]
>>> pascalLine(7)
[1, 7, 21, 35, 35, 21, 7, 1]
```

2. The Levy curves, like the Koch curves, are fractal curves that can be generated recursively. Write a function `levy` that produces a code that can be used to draw the nth Levy curve. Each turn should be 45 degrees. You only need to generate the code, you don't need to draw the curves (though, if you like, you can use the draw function to do so)



Sample runs:

```
>>> levy(0)
'F'
>>> levy(1)
'LFRRFL'
>>> levy(2)
'LLFRRFLRRLFRRFLL'
>>> levy(3)
'LLLFRRFLRRLFRRFLLRRLLFRRFLRRLFRRFLLL'
>>>
```

3. Write a **recursive** function `base` that has two parameters, n, a base 10 positive integer, and b, an integer between 2 and 9. The function returns the base b representation of the number n. The base b representation of a number uses the digits 0,..,b-1 and the place of the digits indicate powers of the base. For example:

   887 (base 10) = 2*7**3 + 4*7**2 + 0*7**1 + 5*7**0 = '2405' (base 7)

   Fortunately, this expansion means that is easy to compute using the % and // operators with respect to the base. For example, suppose that we want the base 7 representation of 887. Then, the last digit is '5' because 887%7=5. And the preceding digits are '240' because 887//7 = 126, and the base 7 representation of 126 is '240'. You may need to conduct additional research on base b representations.

   Sample runs:

```
>>> base(0,3)  # write 0 in base 3
'0'
>>> base(5,3)  # write 5 in base 3
'12'
>>> base(887,7)  # write 887 in base 7
'2405'
```