# Discussion 6: Intro to Recursion

## Factorials are Factorials Times Factorials

**Factorials are defined as the product of a positive integer and all consecutive smaller positive integers. For example, factorial(5) = 5 x 4 x 3 x 2 x 1. Fill in the code below to recursively compute a factorial. Don't worry about the case of n < 1.**

```
factorial(n):
    if_____:
            report _____
    else:
            report _____
```

## PalindromeemordnilaP

**A palindrome is a word that is spelled the same way forwards and backwards. In other words, the first letter must equal the last letter, the second letter must equal the second to last letter ... etc. For the purposes of this problem, all zero-letter and one-letter words are palindromes.**

**Fill in the progression of calls to: is (racecar) a palindrome?**

```
is (racecar) a palindrome?
    is (_____) a palindrome?
            is (_____) a palindrome?
                    is (_____) a palindrome?

                                    _____
```
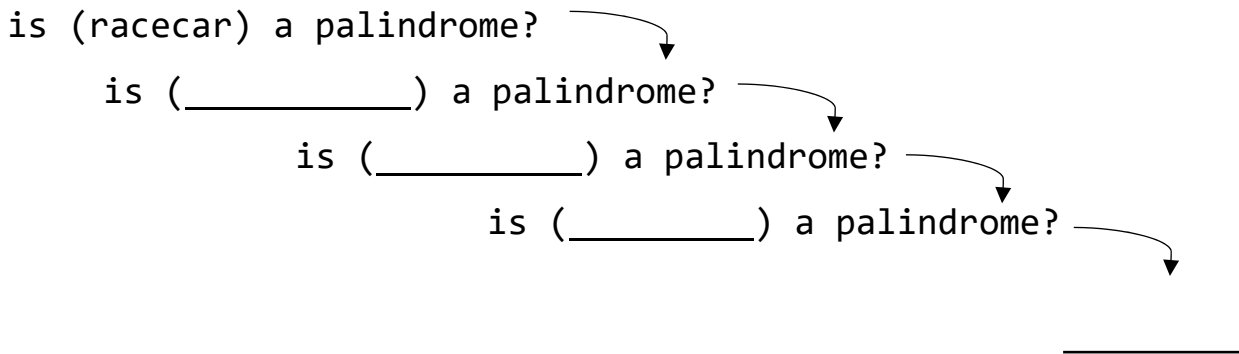
**Using the above information, fill in the recursive palindrome function. You have access to the two functions below.**

```
all-but-first-letter-of( word )
      reports word with the first letter removed
all-but-last-letter-of( word )
      reports word with the last letter removed
```

is (word) a palindrome?:

    if_____:

        report _____

    else:

        if_____:

            report _____

        else:

            report _____

# Where are These Cats Coming From?!

In the following exercise, we will address how to construct and how to think about fractals recursively. You may assume that the sprite starts off at the leftmost part of each level, facing right. *Note that each level is 1/3 of the size of the previous level.*



Level 1:

Level 2:
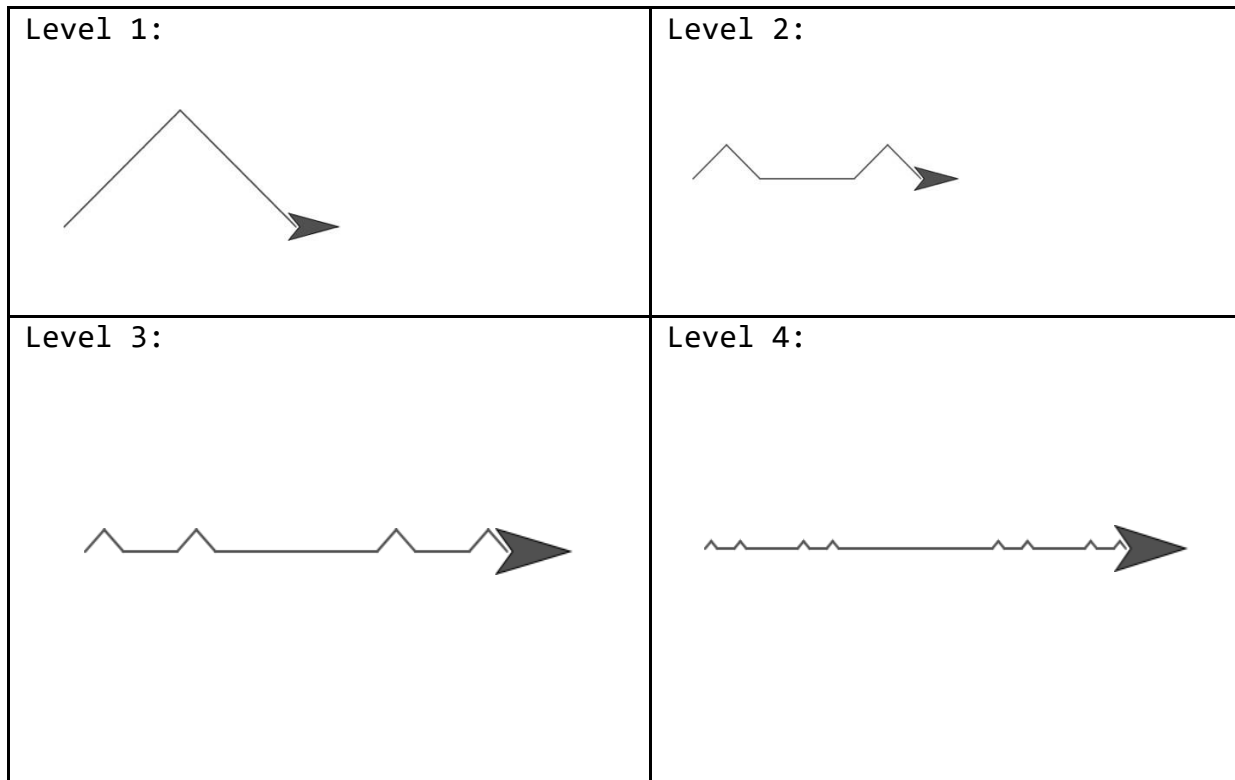
Level 3:

Level 4:

**1) Which level corresponds to the base case?**


**2) For our base case, in what direction does our sprite start and end in?**


**3) In each level, circle each instance of the previous level. Each of these instances refers to one recursive call.**


**4) What does the sprite do between each of the recursive calls?**

## Challenge Problems

**1) Write the function "Boring Multiply," which takes as input a number and a list, and recursively multiplies every item of the list by the number. It should output a _new_ list containing the multiplied values (in order) without modifying the input list.**

| | |
|---|---|
| 1 | **2** - |
| 2 | **16** - |
| 3 | **60** - |
| ⊕ length: 3 |

`Boring Multiply 2 ( list 1 8 30 ◀▶ )`

```
Boring Multiply:
    if_____:
            report  _____
    else:
            report  _____

            _____

            _____

            _____
```

**2) Now the real fun begins. Write the function "Index Multiply," which takes as input a list, and recursively multiplies every item of the list by its index (i.e., position) in the list. It should output a _new_ list containing the multiplied values (in order) without modifying the input list. This problem is harder than it may seem at first glance…don't be afraid to think creatively!**

| | |
|---|---|
| 1 | **1** - |
| 2 | **4** - |
| 3 | **9** - |
| 4 | **44** - |
| 5 | **35** - |
| ⊕ length: 5 |

`Index Multiply ( list 1 2 3 11 7 ◀▶ )`

```
Index Multiply:
    if_____:
            report  _____
    else:
            report  _____

            _____

            _____

            _____
```

**Below are some blocks that you may find useful in writing your solutions to the challenge problems.**

| Block | Description |
|---|---|
| all but first of ▤ | Reports a new list containing all items of the input list, except the first item. |
| all but last of ▤ | Reports a new list containing all items of the input list, except the last item. |
| ▯ in front of ▤ | Appends the input item to the front of the input list, and reports this combination as a *new* list. |
| length of ▤ | Reports the length of the input list. |
| item ⬇ of ▤ | Retrieves the given item of the input list. The "item" variable may be set to any number, "last," or "random." |
| append ▤ ▤ ◀▶ | Appends the two (or more) input lists into a single aggregate list, reporting the output as a *new* list. |
| empty? ▤ | Reports whether the input list is empty. |