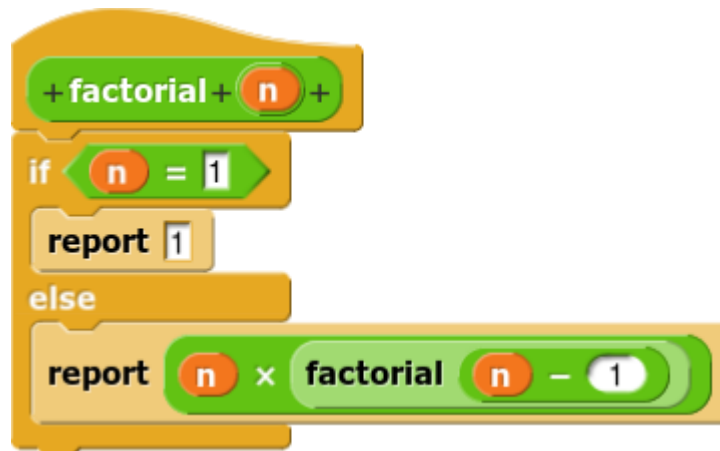


Discussion 6: Intro to Recursion SOLUTIONS

Factorials are Factorials Times Factorials

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

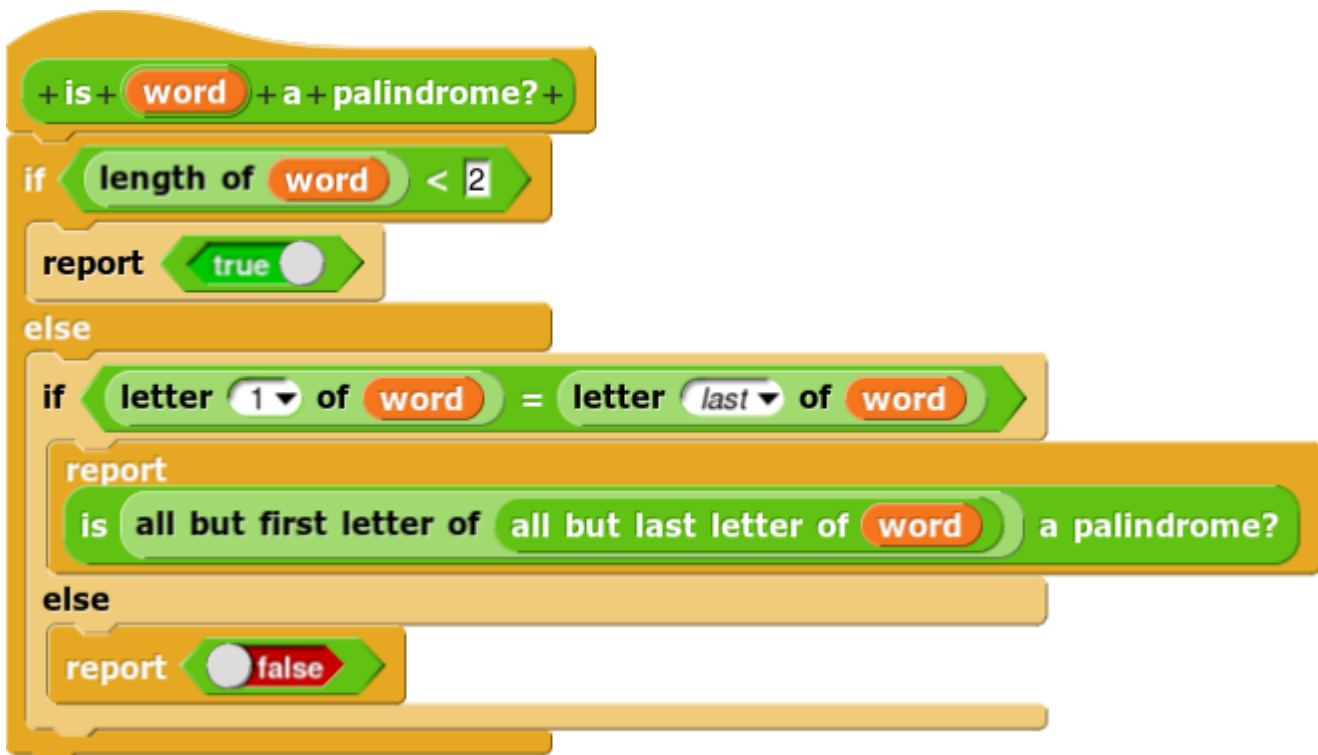


PalindromeemordnilaP

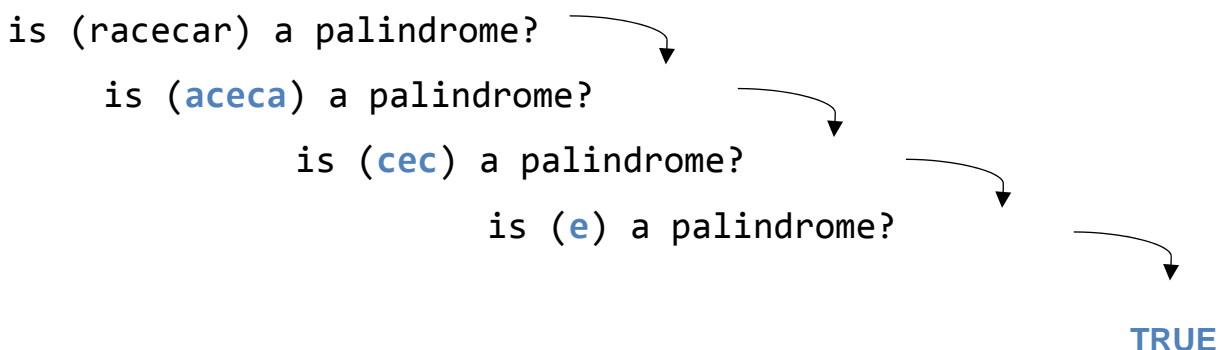
(a) A palindrome is a word that is spelled the same way forwards and backwards. In other words, the first letter must equal the last letter, the second letter must equal the second to last letter ... etc. For the purposes of this problem, all zero-letter and one-letter words are palindromes.

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 gone
all-but-last-letter-of(word)
reports *word* with the last letter gone

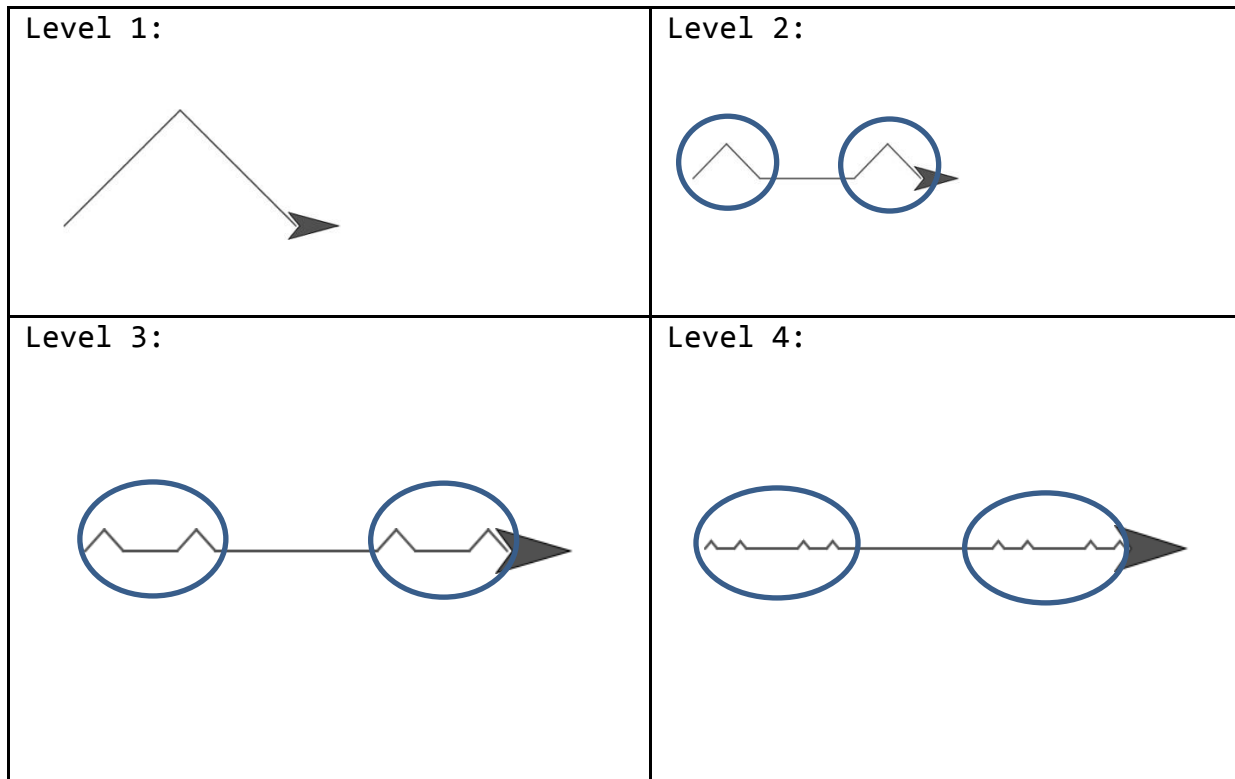


(b) Fill in the progression of calls to: is (racecar) a palindrome?



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.*



1) Which level corresponds to the base case?

Level 1

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

Right (90°)

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

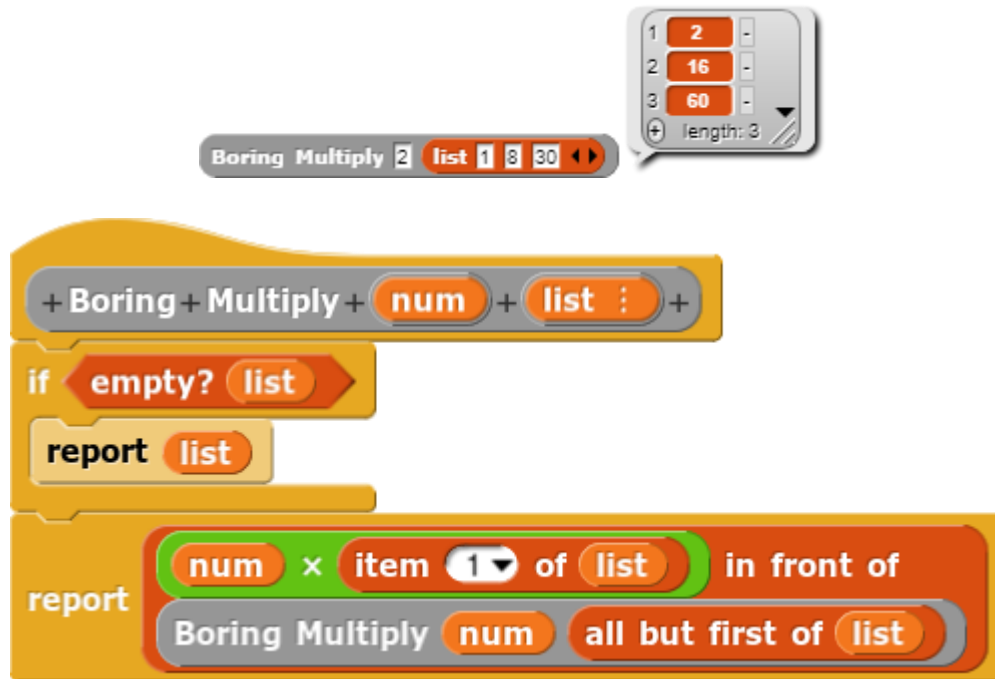
See above

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

Draws a horizontal line

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.



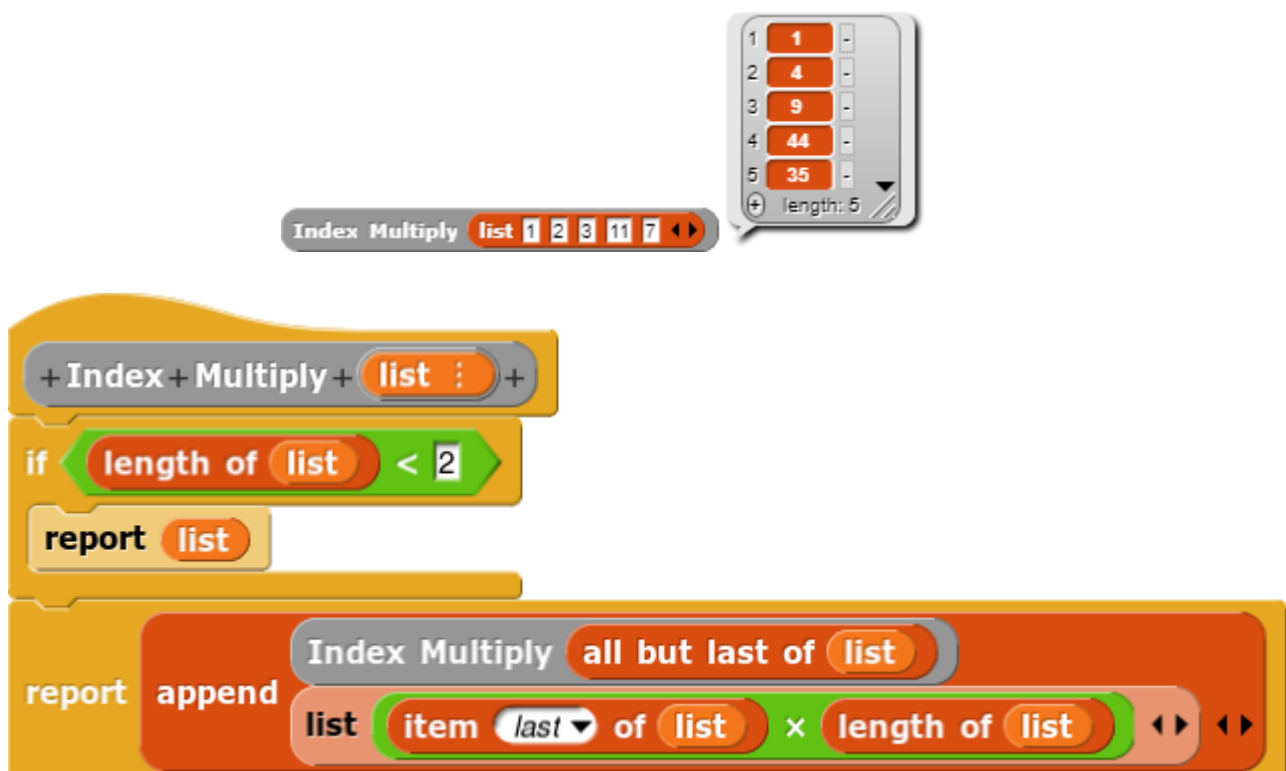
The Scratch code for "Boring Multiply" is as follows:

- Function name: Boring Multiply
- Parameters: num, list
- Logic:
 - if empty? list
 - report list
 - report num × item 1 of list in front of Boring Multiply num all but first of list

Example input: num = 2, list = [1, 8, 30]

Example output: [2, 16, 60]

- 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!










The Scratch code for "Index Multiply" is as follows:

- Function name: Index Multiply
- Parameters: list
- Logic:
 - if length of list < 2
 - report list
 - report append Index Multiply all but last of list list item last of list × length of list

Example input: list = [1, 2, 3, 11, 7]

Example output: [1, 4, 9, 44, 35]

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

Block	Description
	Reports a new list containing all items of the input list, except the first item.
	Reports a new list containing all items of the input list, except the last item.
	Appends the input item to the front of the input list, and reports this combination as a <i>new</i> list.
	Reports the length of the input list.
	Retrieves the given item of the input list. The “item” variable may be set to any number, “last,” or “random.”
	Appends the two (or more) input lists into a single aggregate list, reporting the output as a <i>new</i> list.
	Reports whether the input list is empty.