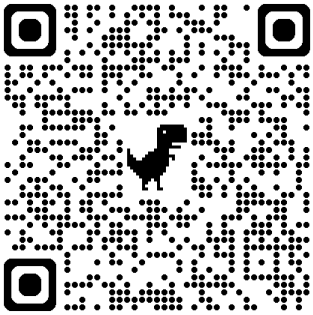
UNIVERSITY OF CALIFORNIA, BERKELEY
Department of Electrical Engineering and Computer Sciences
Computer Science Division

CS10 Fall 2025 **TA: Victoria**



Discussion 7: Linear Recursion + Fractals

Instructions:

- If you're attending this section in-person, please log into iClicker!
- If you missed this discussion, fill out this entire worksheet, and upload it to the Gradescope assignment titled "Discussion 7" by next Discussion.
- Please open up snap.berkeley.edu/run on your computer.
- For the worksheet, you can either explain the process in words, show a screenshot, or draw the block/process.
- Fill out the feedback form by scanning the QR code or going to: **tinyurl.com/fa25-disc-form**

Group Activity / Question of the Day

- In groups of four ask, have you ever pulled an all-nighter to study for an exam or finish a project / homework? On average, how many hours do you sleep?

Required (Pages 2 - 4):

Section I - Linear Recursion

---

1. Recursively add all numbers from 1 to “n”, where “n” is based on the input. It should work like the following:

2. What if I wanted to add all only the even numbers from 2 to n, where “n” is based on the input? It should work like this:

3. Recursively sum the squares of the numbers from 1 to n. It should work like this:

4. ***Recursively*** implement the following versions of the function *glorpify*.

   a. **ARG** will be a word — return the word, but with each character appearing

twice in place. For example, if **ARG** is **“Hello!”**, return **“HHeellllloo!!.** Hint: You need to use the “all but first letter” and “join” functions. You will need to download the “words, sentences” library.

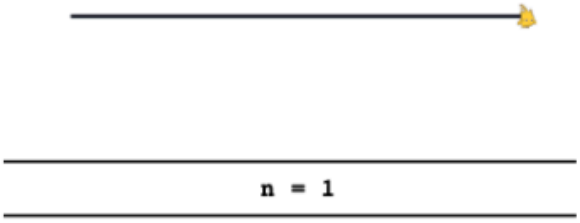
b. **ARG** will be a list of booleans — return **False** if at least one of its items is **False**, and **True** otherwise.

5. Recursively find the number of occurrences of a value in a list. In other words, return the number of times a value appears in a list. It should work like the following:

Hint: You will find the ‘all but first of _” function helpful.

Section II - Fractal

1. Based on the recursing images, create the code for the fractal that will produce the image at each level. You can assume that the angles are always 60 degrees and each recursive call length is ⅓ of the original.



n = 1

n = 2

n = 3

n = 4

Optional Section (Extra Practice):

Optional Section I - Linear Recursion

---

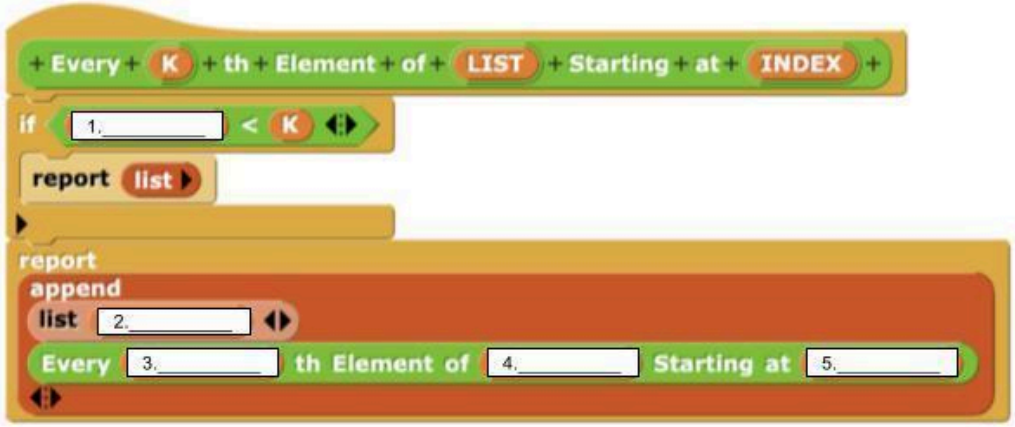
1. Recursively report a new list that filters out any odd numbers. The list returned should only return even numbers in the list. It should work like the following:

2. Report true if a word is a palindrome. A palindrome is spelled the same backwards as it is forwards.Examples: RACECAR, LEVEL, RADAR, CIVIC, A, B, C, … It should function like the following:

Hint: You will need to download a library in Snap! Called "words, sentences". You will need "all but first letter of" and "all but last letter of"

3. Write a procedure that takes in a list **LIST**, a positive integer **K**, and a positive integer **INDEX.** It should return a new list containing every **K**-th element of **LIST** starting at (but not including) **INDEX**. It should function like the following:

Fill out the blocks labeled '**ANSWER-X**' in the skeleton code:



Optional Section II - Fractals

1. Based on the recursing images, create the code for the fractal that will produce the image at each level. Each level of the fractal is ½ the size of its parent level. For example, in the n=2 figure above, each line is ½ the length of the line from the n=1 level.