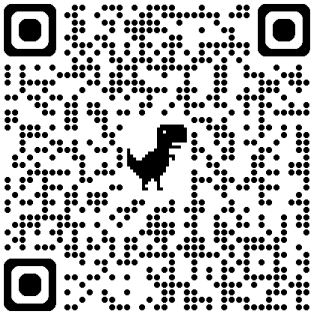
UNIVERSITY OF CALIFORNIA, BERKELEY
Department of Electrical Engineering and Computer Sciences
Computer Science Division

CS10 Spring 2025 **TA: Victoria**



Discussion 5: Recursion + Nested Lists

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 5" 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.

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):

Assigned Reading

Do you think most people are aware of how much data they generate and share daily? Why or why not? Can you think of an example where someone's digital footprint (e.g., an old post, email, or metadata) had / might have serious consequences? Should governments or companies be allowed to track and collect metadata from individuals? Why or why not?

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. ***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.

3. 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 - Nested Lists

1. Create a sum function that takes in a nested list and outputs the sum of all the values. It should function like this:

2. Create a function that adds a new column. The new element should be the sum of that particular row. It should function like this:

Since row 1 contains: 1, 5. The new row contains: 1, 5, 6 (where 6 is the sum of 1 and 5).

Optional Section (Extra Practice):

Optional Section I - Linear Recursion

---

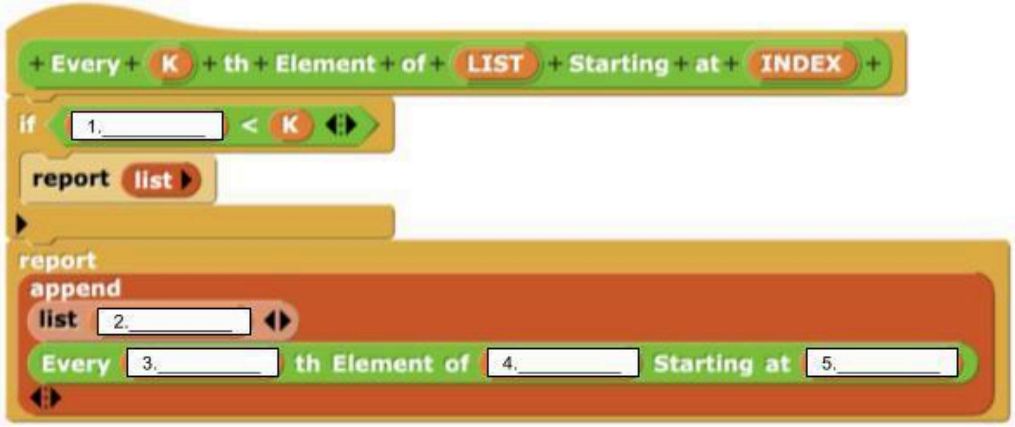
1. Recursively report a new list that filters out any odd numbers. The list returned should only return even numbers in the list. It should work like the following:

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 - Nested Lists

---

1. Create a function that finds the maximum number in the nested list using iteration. It should function like this:

2. Do the same function, but find a solution only using HOFs!