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

CS10 Summer 2024

TA: Wen Cao and Naveen Nathan

Discussion 5: Discussion 5: Discussion 5: Discussion 5: Discussion 5: Discussion 5: RECURSION 🥰

Instructions

- If you're attending this section in-person, at the end of section, you'll be given a secret phrase. Please enter it here: ***tinyurl.com/cs10su24***
- If you're attending asynchronously, you must fill out this entire worksheet, and upload it to the Gradescope assignment by the next discussion section.
- Please open up *snap.berkeley.edu/run* on your computer.
- Ask the person to your right / left if they'd rather live in a house that's always a bit too hot, or a house that's always a bit too cold.

Q1. Recursion: Lingua Franca

A. What is *recursion*? What does it mean for a procedure to be defined *recursively*?

B. What are the two main components that comprise the anatomy of a recursive procedure?

C. What are the differences between the two cases, and what are the connections between them? What happens if we don't reach the base case?

D. What is *tree recursion*? What does it mean for a procedure to be defined *tree recursively*? What is the other form of recursion we've studied?

Q2. Implementing Recursive Procedures: Accumulation

Recursively implement the following versions of the function *glorpify*, which return the result of accumulating the single input **`ARG`** according to the specification.

Version-1 (interactive)

`ARG` will be a list of numbers — return the sum of the items of **`ARG`**.

Version-2

`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.

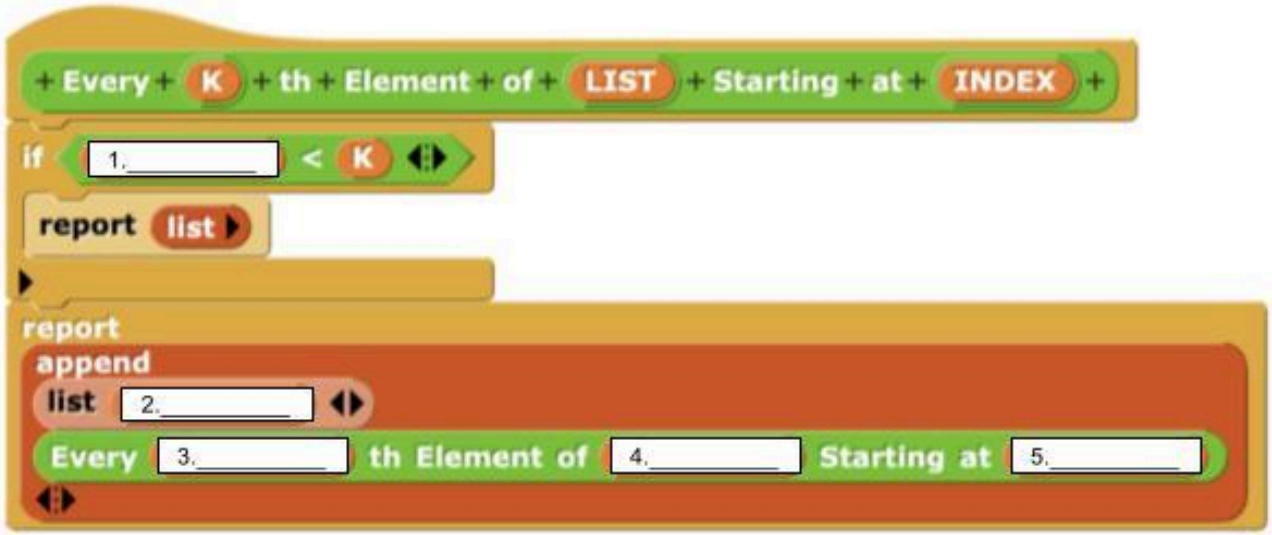
OPTIONAL EXERCISE: Version-3

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

Q3. Linear Recursion: Every k element

Write a procedure that takes in an input **LIST**, positive integer **K**, and a positive integer **INDEX** (which is always **<=** the length of the list), and returns a new list containing every **K**-th item of **LIST**, starting at (but not including) **INDEX**, in order. Here are some examples:

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



1. _____

2. _____

3. _____

4. ____

5. ____

Q4: Tree Recursion: Combinations

Given a list of positive numbers L and a target positive number N, write a reporter that generates all the possible subsets of L that sum up to N. It should output a list of lists, wherein each inner list contains a subset of numbers from L that sum to N.

Can you write this procedure using both tree and linear recursion? Which version is more efficient? Can you write it in a manner such that its runtime is better than exponential?

Hint: Here's a naive approach — first, generate all possible subsets of L, and then filter out those that don't sum to N.

Q5. OPTIONAL EXERCISE: Tree Recursion: <u>Hop-Hop</u>

1. We're climbing a staircase with a total of N steps. We can take either 1 or 2 steps at a time (a mix of both is possible) to reach the top. Write a procedure that takes in N and returns the number of ways that we can climb the stairs.

For example, if N = 3, then you can climb the stairs in the following ways:

Take 1 step, 1 step, 1 step
Take 1 step, then 2 steps
Take 2 steps, then 1 step

So, the result is = 3.

2. Now, we're getting ambitious and decide that we can climb up to K steps at a time. (In the previous problem, K = 2.) Write a procedure that takes N and K as inputs and returns the number of ways that we can climb the stairs.