Lab 10: Recursive Reporters

Instructions:

This worksheet serves as a guide and set of instructions to complete the lab.

- You must use the starter file, found here, to get credit for the lab.
- Additionally, here is the workbook that you can read through for further context and additional (non-required) material.
- All material was sourced from the CS10 version of The Beauty and Joy of Computing course.

IMPORTANT:
Be sure to use the starter file linked here and not the one linked in the workbook! Otherwise, you will fail the autograder and not get credit for the lab.

Submitting:

You will need to fill in the blocks under the starter file and submit to Gradescope.

- To receive full credit, you will need to complete the required blocks, and the required blocks must pass all tests from the autograder in Gradescope.
- For instructions on how to submit to labs to Gradescope, please see this document.

Please note, you must use the starter file, and you must NOT edit the name of any of the required blocks. Failing to do either for these will result in the autograder failing.

The name of the file may be incorrect. Please use the links provided.

Objectives:

Recursion is arguably the second most important topic in this course (after abstraction of course). In today's lab you will implement what you know about recursion into reporter blocks. By the end of the lab, you will:

- Understand how reporters use combiners instead of sequences.
- Practice planning and coding recursive reporters.
- Implement higher order functions in non iterative solutions

Required Blocks:

** It is highly recommended that you work with a partner for this assignment.

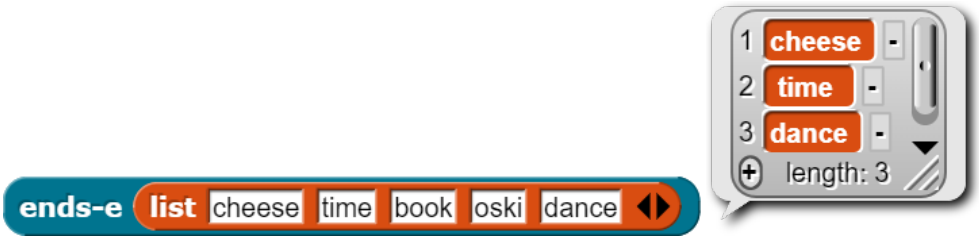
- Block 1: ends-e (input)
- Block 2: numbers (input)
- Block 3: subset sum (lst, target)

Important Topics mentioned in the Workbook:

For better understanding of the lab we highly recommend going through these workbook pages! Topics that are important but not required for this lab will be indicated with an asterisk**. These topics are best reviewed in order and as you complete the lab.

- Recursive Reporters
- Plurals of Words
- What Is Pascal's Triangle?
- Pascal's Triangle
- How Long Does This Computation Take? **
- Even Numbers
- Sorting a List + Cont **
- Subsets of a Set + Subsets and Efficiency
    - This portion of the workbook helps explain the concept of (sub)sets
    - However it does NOT explain the steps to complete block 3

Block 1: ends-e (input)

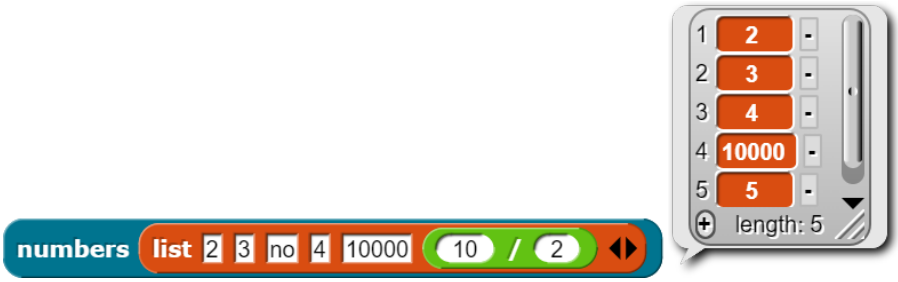
- Objective:
    - Edit a reporter block, ends-e, that takes a list of words as input, and reports a list of those words from the input whose last letter is e
- Note
    - You must recursion, no HOFs
- Inputs:
    - Input = anything
        - Input should be a *list* of words
- Output:
    - Reports: a list
        - A list if all the words from the list that ends with e
        - If there are no words that end with e, the reporter reports an empty list
- Examples:
    - 

    - 

Block 2: numbers (input)

- Objective:
  - Edit a reporter block, numbers, that takes a list of mixed words and numbers as input, and reports a list of just the numbers from the input list.
- Notes:
  - You must use recursion. No HOFs.
- Input = anything
  - Input should be a *list* of words, strings, and numbers
- Output:
  - Reports: a list
    - A list if all the items in the list that are numbers
    - If there are no numbers in the list, the reporter reports an empty list

- Examples:
  - 

  - 

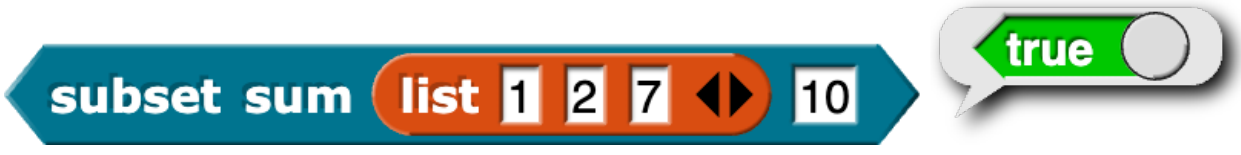
  - 

Block 3: subset sum (lst, target)

- Objective:
  - Write a function subset_sum(lst, target) that takes a list of integers and a target integer. The function should return True if there exists a subset of lst that sums to exactly target, and False otherwise.

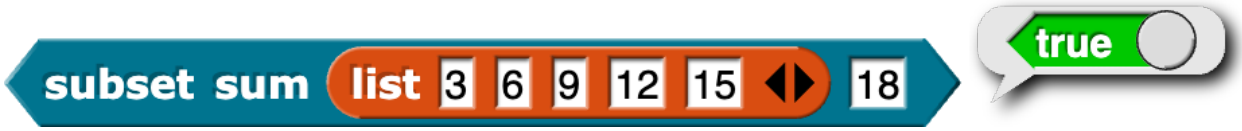
- Notes:
  - You must use recursion. No HOFs and no iteration.
  - *Hints:*
    - *This problem can be solved recursively by either including or excluding each element in the list:*
    - *Try including the first element in the subset and reduce the target by the value of that element.*
    - *Try excluding the first element from the subset and leave the target unchanged.*
    - *Recursively continue until the target becomes 0 (indicating that a subset was found) or there are no elements left.*
- Inputs
  - Lst = a list of numbers
  - Target = a number
- Output:
  - Reports: boolean

- Examples:
-

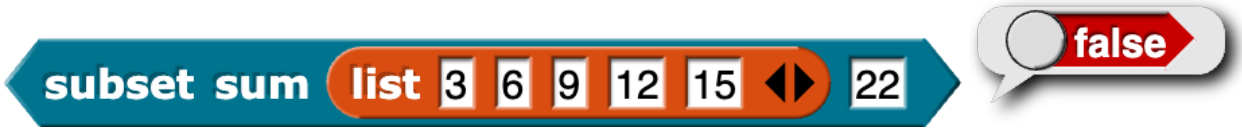


-

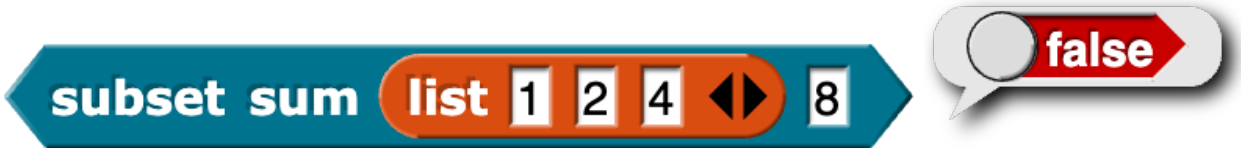
-
-



-



-



You can always check the validity of your solutions by using the local autograder. Remember to submit on Gradescope!