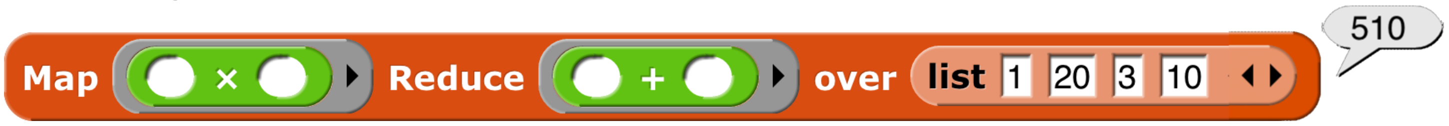
CS10 With-Computer Final (Summer 2018, Sec 1)

There are three questions, two Snap*!* ones and a Python one. Save your Snap*!* code into a Snap*!* file, and name it **`FinalYourfirstnameYourlastname.xml`** (e.g., **`FinalAlanTuring.xml`**). For the Python question, create a new Python file and name it **`FinalYourfirstnameYourlastname.py`** (e.g., **`FinalAlanTuring.py`**). Submit both files on bCourses under the "online" final assignment for your lab section. All questions are independent.

Snap*!* Questions: (use this starter file: https://bjc.link/2Mhophy**)**

MapReduce is a programming model utilized in cluster computing, most famously by Google to reindex the World Wide Web. The general idea behind MapReduce is that you want to map some function over some collection of data, and then you want to use another function to reduce it down into one value. Take the following example:



Here, the multiply operator is mapped over each item of the list (producing a list of 1, 400, 9, 100), and the addition operator reduces all of that list into one number, 510 (1 + 400 + 9 + 100). For this problem, we want you to explore a couple of different ways of implementing the MapReduce block.

a) (3 pts) Write MapReduce *only using higher-order functions* (i.e., only **`map`**, **`keep`** and/or **`combine`**). Write your solution in the **`MapReduce(HOF)`** block from the starter file.

b) (6 pts) Write it *recursively*. Write your solution in the **`MapReduce (recursive)`** block from the starter file. You may not use any iteration (**`repeat`**, **`repeat until`**, **`for`**, **`for each`**) or higher-order functions in this solution. You may use **`call`** and **`run`**. You can assume that there will always be *at least one* item in the list.

Python Question (6 pts):

Write a function that returns the *longest list of unique words that start with the same letter* given any string of lower-case words. You *must* use a dictionary in your solution; if you forget any commands, remember there's **`help(`***type***`)`** and **`dir(`***type***`)`**, as in **`help(dict)`** or **`dir(str)`**. If a word is repeated in a string, you should not include that word again. In the case of a tie, you may return any of lists that correspond to the letters that have the most words in that string. In other words, if there were an equal number of words that start with 's' and 't', you can return either the list of words that start with 's' or the list of words that start with 't'. You may find the **`min`** and **`max`** functions helpful.

```
starts_with_same_letter("to be or not to be that is the question")➔
['to', 'that', 'the']
starts_with_same_letter("been there believe that")➔
['been', 'believe']
```