

Discussion 12: In-Lab Final Review

Snap Practice

1. For this problem, you will be writing the block `vowels in`, which takes in a word/sentence, and reports the number of vowels in it.

`vowels in` CS10 is cool! 3

a. First, write this function using only HOFs (map, keep, and combine).

```
+ vowels in + phrase +
report
  length of
    keep items such that
      list a e i o u contains
    from
      split phrase by letter
```

b. Now, write this function using only recursion. You may not use any loops.

```
+ recursive vowels in + phrase +
if length of phrase = 0
  report 0
if list a e i o u contains letter 1 of phrase
  report 1 + recursive vowels in all but first letter of phrase
else
  report recursive vowels in all but first letter of phrase
```

2. Sometimes, a word that isn't a palindrome will have palindromes in it. For example, the word "ballad" has two palindromes in it: "alla" and "ll." Use recursion to write a function called `subpalindromes` that takes in a word, and returns the number of palindromes contained in it.

For this problem, you only need to find the smaller palindromes whose centers are found in the middle of the word. For example, "cc" in the word "accel" would not count as a sub-palindrome.

```
+ subpalindromes + word +
if length of word = 0
  report 0
if length of word = 1
  report 1
else
  if is word a palindrome?
    report
      1 +
      subpalindromes all but first letter of all but last letter of word
  else
    report
      subpalindromes all but first letter of all but last letter of word
```

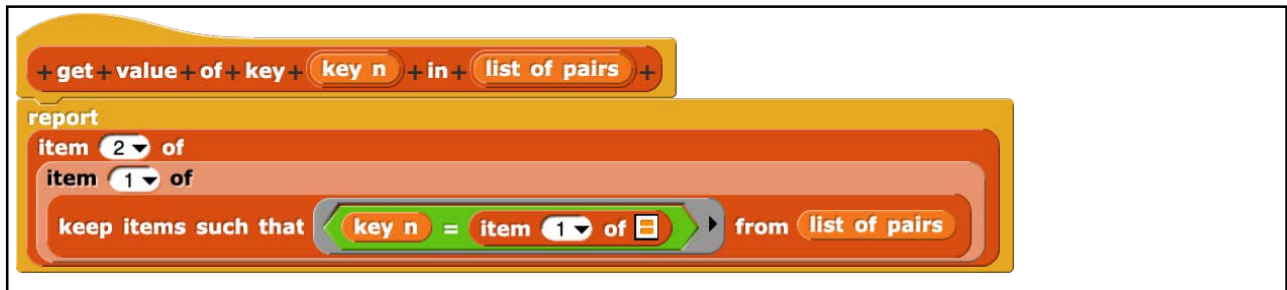
3. For this problem, we will be representing key-value pairs in Snap!. Our key-value pairs can be represented as follows, where key 1 corresponds to value 1.



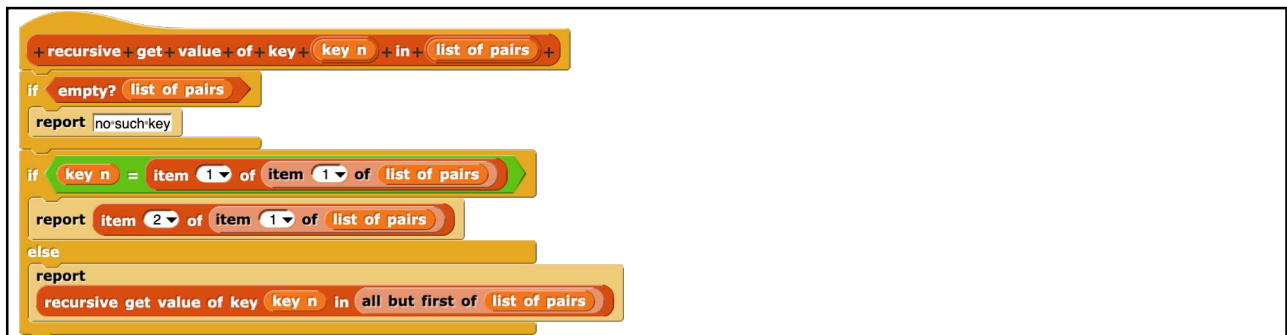
We would like to write a function, `get value`, that takes in a list of key-value pairs and a key and returns the value corresponding to that key.



a. First, write this function using only HOFs (map, keep, and combine).



b. Now, write this function using only recursion. You may not use any loops.



Python Practice

1. Write a function called `dictionary_reverser` that takes in a dictionary and returns a new dictionary with the original values as the keys and the original keys as lists of values.

```
>>> dictionary_reverser({1: 3, 2: 3, 8: 9})
{3: [1, 2], 9: [8]}
```

```
def dictionary_reverser(dict):
    r = {}
    for k in dict:
        if dict[k] in r:
            r[dict[k]].append(k)
        else:
            r[dict[k]] = [k]
    return r
```