CS 331 — Immutable Lists Activity

Mattox Beckman

Introduction

You saw some functions in the video that manipulated immutable lists. Now you get to write some of your own.

Questions

```
(defn my-length [xx]
   (if (nil? xx)
      0
      (+ 1 (length (:rest xx)))))
```

Figure 1: Length

1. What is the time complexity of the `length` function above? Can it be done in $\mathcal{O}(1)$ time? Why or why not?

2. Using hashmaps, we can use a list as a dictionary[1]. Suppose a list has elements of the form `{:key X :value Y}`. Write a function `find` that takes such a list and a key and returns the corresponding value if the key is in the structure. Return `nil` if it's not in there.

```
user> (def phonebook (mklist {:key "emergency" :value "911"}
                             {:key "jenni" :value "867-5309"}))
#'user/phonebook
user> (find "jenni" phonebook)
"867-5309"
user> (find "billy" phonebook)
nil
```

[1] Such a list is called an *associative list*.

3. Write a function `add` that adds a new key and value to the associative list.

```
user> (def p' (add "empire" "588-2300" phonebook))
#'user/p'
user> (find "empire" phonebook)
nil
user> (find "empire" p')
"588-2300"
```