CSCI 334: Principles of Programming Languages

Lecture 10: Functional Programming

Instructor: Dan Barowy

Williams

Announcements

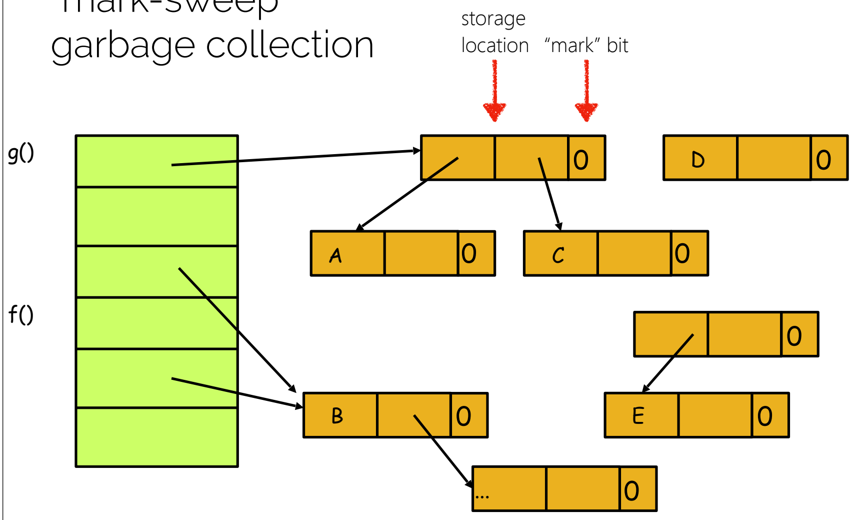
Midterm exam *next class*

You should have feedback for all HW—
if not, please let me know!

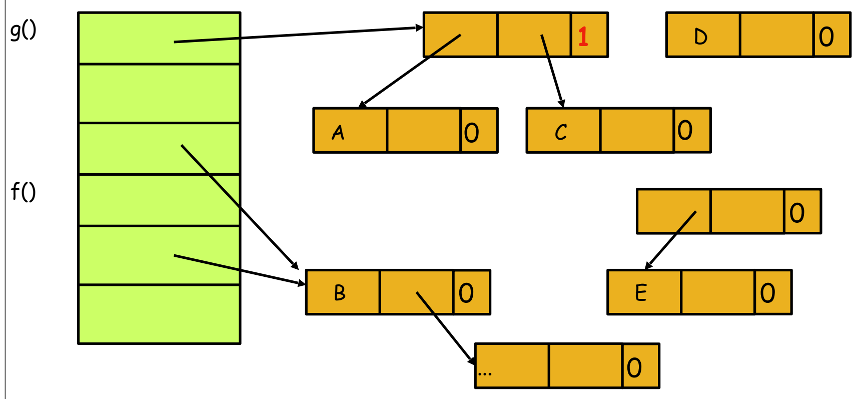
"Recursive Functions [...]" (McCarthy)

Lisp	C
`car`	`head`
`cdr`	`tail`
`cons`	`prepend`

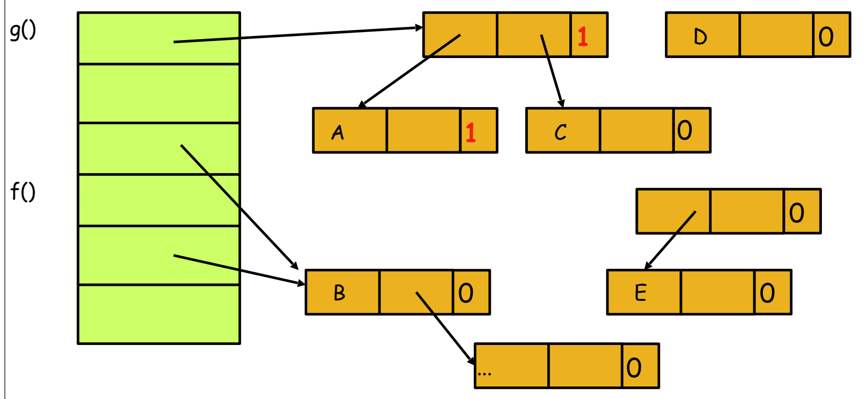
"mark-sweep" garbage collection



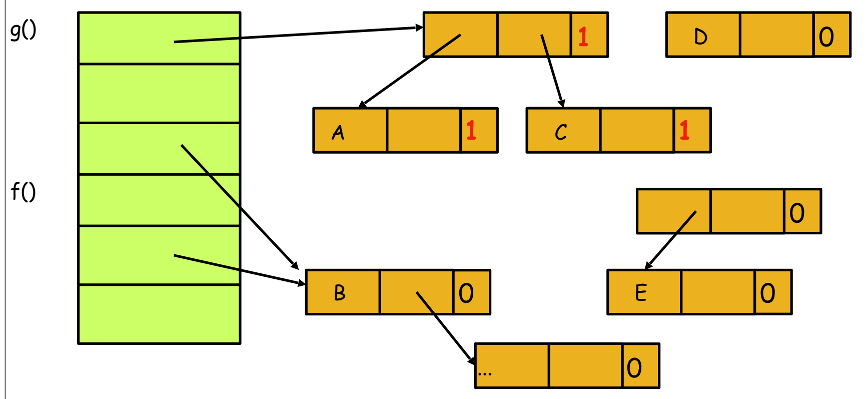
1. Mark reachable cells



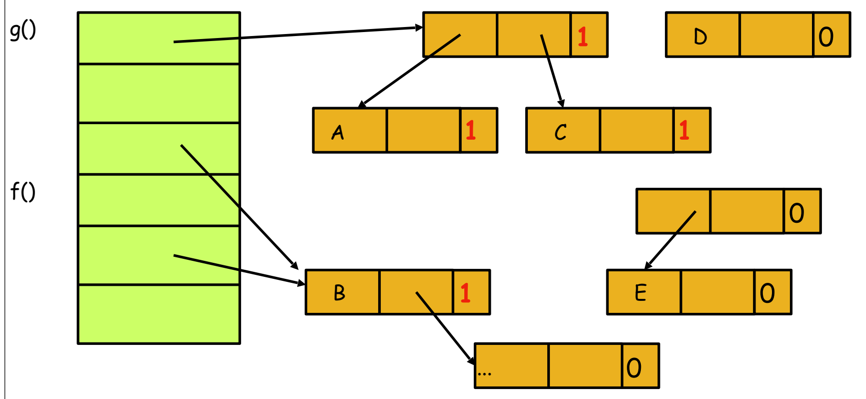
1. Mark reachable cells



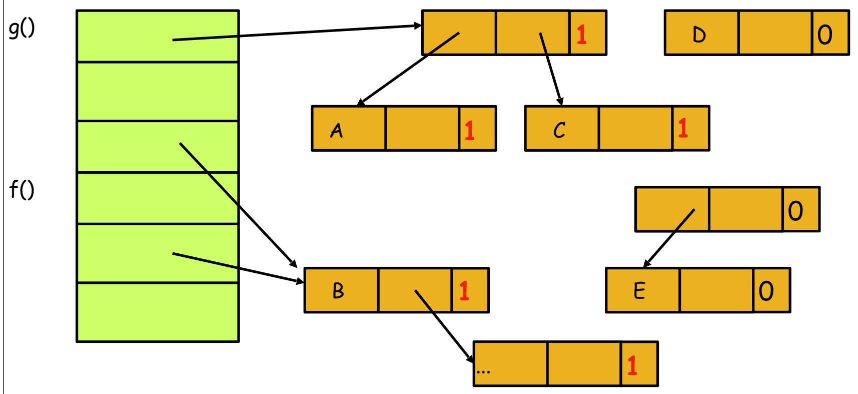
1. Mark reachable cells



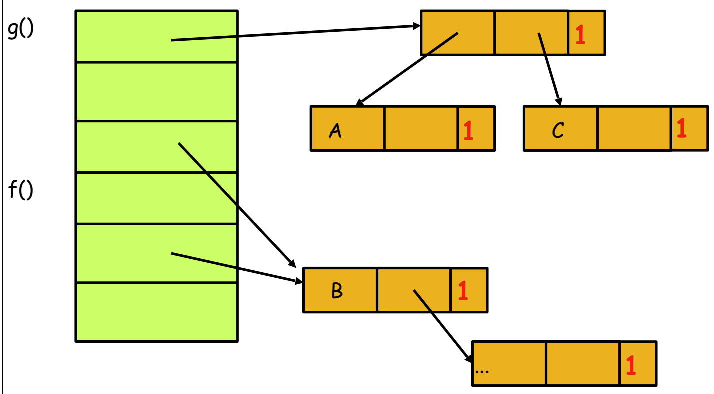
1. Mark reachable cells



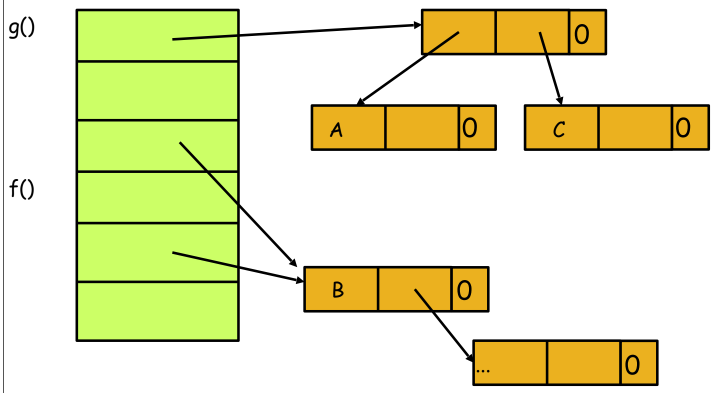
1. Mark reachable cells



2. Free ("sweep") unreachable cells



3. Clear tags



Activity

list length

`(length-list '(1 2 3 4 5 6))` ↠ 6

Mental technique #4

“Growth” mindset

“In a fixed mindset students believe their basic abilities, their intelligence, their talents, are just fixed traits. They have a certain amount and that's that, and then their goal becomes to look smart all the time and never look dumb. In a growth mindset students understand that their talents and abilities can be developed through effort, good teaching and persistence.”

— Carol Dweck (Lewis and Virginia Eaton Professor of Psychology at Stanford University)

Individuals with a "growth" mindset are more likely to continue working hard—and succeed—despite setbacks.

Mental technique #4

“Growth” mindset

Your brain is a machine designed to accommodate to a changing world.

Mental technique #4

Demonstration

Mental technique #4

Demonstration (again)

If that made sense to you, raise your hand.

Mental technique #4

Demonstration (ungarbled)

Mental technique #4

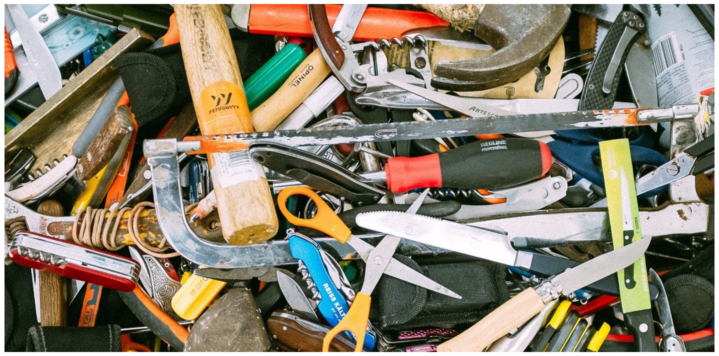
Demonstration

Mental technique #4

Demonstration (again)

Anil Seth, “Your brain hallucinates your conscious reality”

Why am I telling you this?



This course is about priming your brain with different ways of thinking about programming.

Why am I telling you this?

You can be a programmer without these ideas.

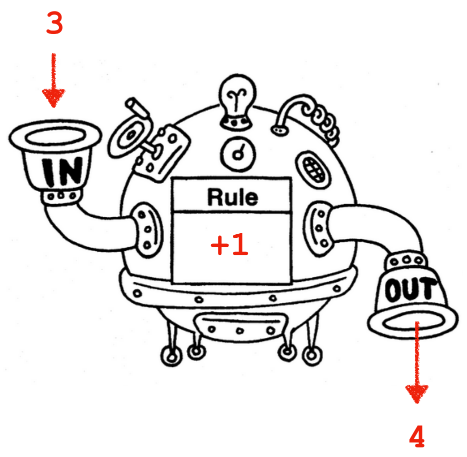
But make the effort to internalize these concepts and you will see their application everywhere.

You will be a *clearer* thinker
and a *better* programmer.

Three amazing concepts from FP

- First-class functions
- Higher-order functions
  - map
  - fold

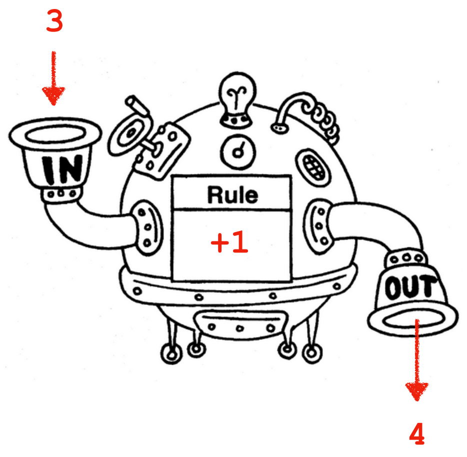
a function



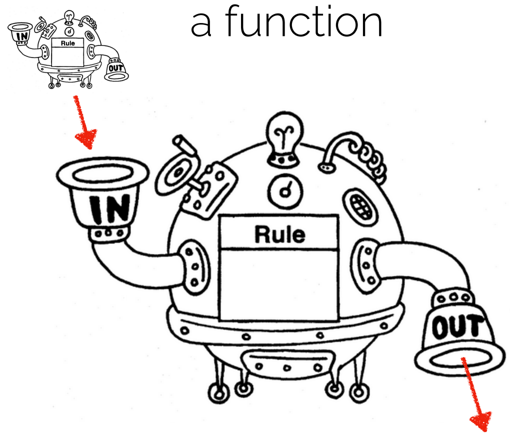
"first class" function

Functions are values in a programming language

a function

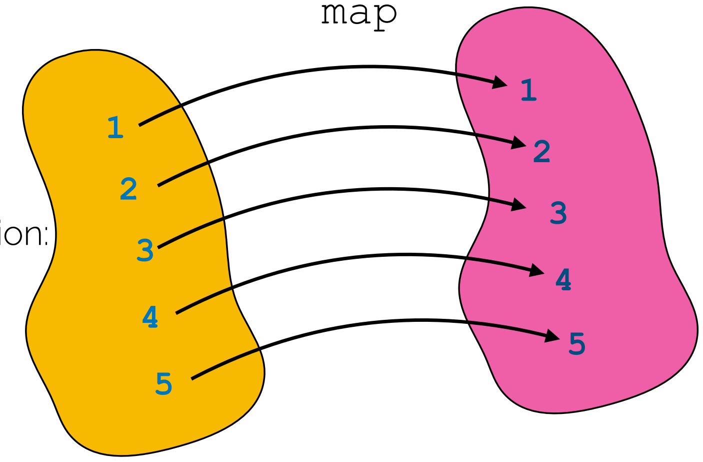


a function



map

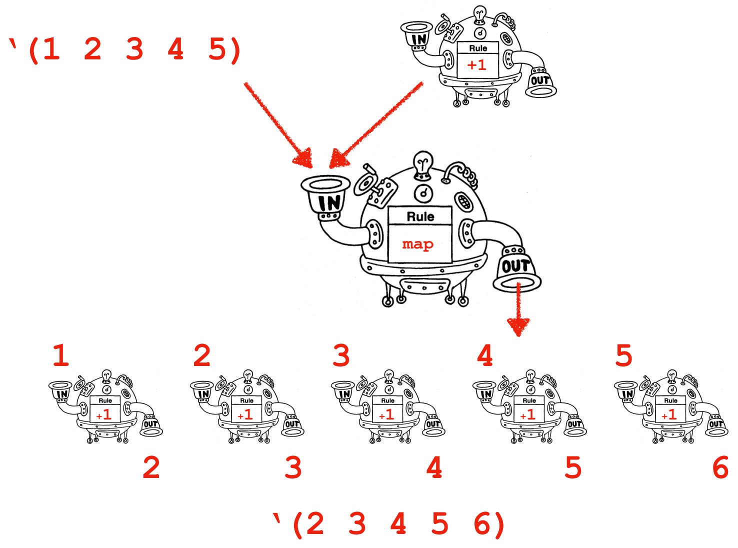
Intuition:



Like a `for` loop, but without mutable variables

```
(mapcar (lambda (x) (+ x 1) '(1 2 3 4 5))
```

map



fold

Intuition:

fold left

```
(reduce #'+ '(1 2 3) :initial-value 0)
```

```
acc = 0, `(1 2 3)
acc = 0+1, `(2 3)
  acc = 1+2, `(3)
   acc = 3+3, nil
  returns acc = 6
```

fold right

```
(reduce #'+ '(1 2 3):initial-value 0
                     :from-end t)
```

```
`(1 2 3), acc = 0
`(1 2), acc = 0+3
`(1), acc = 2+3
nil acc = 5+1
returns acc = 6
```

what does this print?

```
(reduce #'append '((1) (8))
    :initial-value '(w i l l i a m s))
```

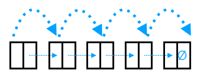
how about?

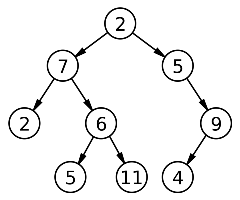
```
(reduce #'append '((1) (8))
    :initial-value '(w i l l i a m s)
    :from-end t)
```

fold

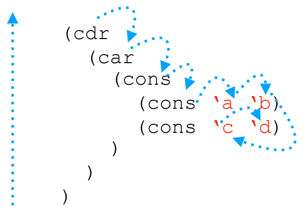
structural recursion ➞ fold it!

(in a nutshell: any problem that recurses on a subset of input)


list length



tree height



```
(cdr
  (car
    (cons
      (cons 'a 'b)
      (cons 'c 'd)
    )
  )
)
```

evaluation