CS61A Lecture 4

2011-06-23

Colleen Lewis

Remove all non-even numbers from a sentence

```
STk>(evens '(2 8 3))
(2 8)
```

```
(evens '( 2 8 3 ))
(se 2
    (evens '( 8 3 ))
    (se 8
        (evens '( 3 ))
        (se
            (evens '())
            '()
(se 2 (se 8 (se '())))
```

Remove all non-even numbers from a sentence

```
STk>(evens '(3 9 6))
(6)
```

```
(evens '( 3 9 6 ))
(se
    (evens '( 9 6))
    (se
        (evens '( 6 ))
        (se 6
            (evens '())
            '()
(se (se (se 6 '())))
```

Write evens

```
;Assume you have the predicate even?
(define (even? x)
    (= 0 (remainder x 2)))
```

A)easy B)medium C)hard D)stuck

evens Solution

```
(define (evens sent)
    (cond
        ((empty? sent) _____)
        ((even? (first sent)) _____)
            (se (first sent)
                (evens (bf sent))))
        (else _____)
            (se(evens (bf sent))))))
```

odds Solution

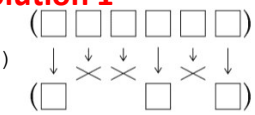
```
(define (~~evens~~ odds sent)
    (cond
        ((empty? sent) '())
        (~~(even?~~ odd? (first sent))
            (se (first sent)
                (~~evens~~ odds (bf sent))))
        (else
            (~~evens~~ odds (bf sent))))))
```

pronouns Solution

```
(define (~~evens~~ sent)
    (cond
        ((empty? sent) '())
        ((~~even?~~ (first sent))
            (se (first sent)
                (~~evens~~ (bf sent))))
        (else
            (~~evens~~ (bf sent)))))
```

pronouns
pronoun?
pronouns
pronouns

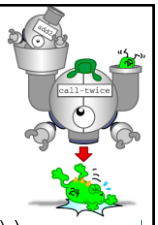
keep Solution 1



```
(define (~~evens~~ sent)
    (cond
        ((empty? sent) '())
        ((~~even?~~ (first sent))
            (se (first sent)
                (~~evens~~ (bf sent))))
        (else
            (~~evens~~ (bf sent)))))
```

keep
pred?
keep
keep

keep Solution 2




```
(define (keep pred? sent)
    (cond
        ((empty? sent) '())
        ((pred? (first sent))
            (se (first sent)
                (keep pred? (bf sent))))
        (else
            (keep pred?(bf sent)))))
```

Calling keep

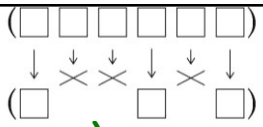
```
(define (keep pred? sent)
    (cond
        ((empty? sent) '())
        ((pred? (first sent))
            (se (first sent)
                (keep pred? (bf sent))))
        (else
            (keep pred? (bf sent)))))

STk> (keep even? '(2 8 3))
(2 8)
```

Calling keep

```
STk>(keep odd? '(1 4 8 5))
(1 5)
STk>(keep pronoun? '(I me not you))
(I me you)
STk>(keep even? '(1 3 5 7))
()
STk> (keep (> 5) '(1 3 5 7))
```

AHHHH! Not a function!!!

keep



(**keep** *procedure sent*)

- *procedure*
  - a procedure that takes in **one argument**
  - a procedure that returns #t or #f
- *sent*
  - a sentence with 0 or more words
  - a word with 0 or more letters

Try it! Make this work!

```
STk>(keep _______________ '(8 3 4 7))
(8 7)
```

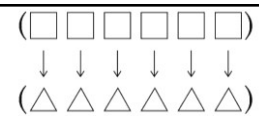
```
(lambda (x) (> x 5))
```

Talk to your partner about why (> 5) doesn't work!!!

Try to write it without a helper method

A)easy B)medium C)hard D)stuck

every



(every *procedure sent***)**

- *procedure*
  - a procedure that takes in **one argument**
  - a procedure that returns a word or a sentence
- *sent*
  - a sentence with 0 or more words
  - a word with 0 or more letters

Try it! Make this work!

```
STk>(every _______________ '(8 3 4 7))
(18 13 14 17)
```

```
(lambda (x) (+ x 10))
```

Talk to your partner about why (+ 10) doesn't work!!!

Try to write it without a helper method

A)easy B)medium C)hard D)stuck

Try it! Define s-g-t-100

Squares-greater-than-100

```
STk>(s-g-t-100 '(2 9 13 16 9 45))
(169 256 2025)
(define (s-g-t-100 sent)
    (keep
        (lambda (x) (> x 100))
        (every
            (lambda (y) (* y y))
            sent)))
```

Using lambda with define

- These are VERY DIFFERENT:

```
(define (cat y)
    (lambda (x) (+ x 1)))

(define dog
    (lambda (x) (+ x 1)))
```

Using lambda with define

```
(define (cat y)
    (lambda (x) (+ x 1)))
(define dog
    (lambda (x) (+ x 1)))
```

Which of these gives the answer 2?

I (cat 1)

II ((cat 1) 1)

III (dog 1)

IV ((dog 1) 1)

Correct Answer

A) I&III

B) II&IV

C) I&IV

D) II&III

E) Not sure