Wrap-Up Quiz

Bookmark this page

Wrap-Up 05.1

1/1 point (graded)

According to the lessons, what mathematical technique is closely related to recursion in programming?

- Incursion
- Induction
- Inflection
- Intersection
- Inversion

Show answer

Submit You have used 1 of 1 attempt

Wrap-Up 05.2

1/1 point (graded)

Which of the following statements is *false*?

- Every structurally recursive function must have at least one base case.
- Every structurally recursive function must have at least one recursive case.
- Every structurally recursive function must have a wrapper function.
- Every structurally recursive function must have a pattern of recursion that mirrors a data definition.
- Every structurally recursive function must include its own name somewhere in its body.

Show answer

Submit You have used 1 of 1 attempt

Wrap-Up 05.3

1/1 point (graded)

Which of the following expressions will evaluate to `false` ?

- `(empty? (cons empty empty))`
- `(cons? (cons empty empty))`
- `(list? (cons empty empty))`
- `(empty? empty)`
- `(list? empty)`

Show answer

Submit You have used 1 of 1 attempt

Wrap-Up 05.4

Define a constant called `my-list`. It should be defined as a list of length 3, containing a string, followed by a symbol, followed by a Boolean (the precise values are up to you).

1/1 points
Attempts: 1 / Unlimited

```
(define my-list (cons "string" (cons 'symbol (cons true empty))))
```

Submit Code Reset Code

Correct! 1/1 points

Wrap-Up 05.5

1/1 point (graded)

Suppose that a constant `L` is defined to contain a list of length n. What is the value of `(length (cons L empty))` ?

- 1
- 2
- n
- $n + 1$
- There isn't enough information to know.

Show answer

Submit You have used 1 of 1 attempt

Wrap-Up 05.6

Write a function `first-negative` that consumes a list of numbers `lon` and produces the first element of the list that's strictly less than zero. If no such element is found, the function produces the symbol `'not-found`.

1/1 points
Attempts: 1 / Unlimited

```
(define (first-negative lon)
  (cond
    [(empty? lon) 'not-found]
    [(>= (first lon) 0) (first-negative (rest lon))]
    [else (first lon)]))
```

Submit Code Reset Code

Correct! 1/1 points

Wrap-Up 05.7

1.0/1.0 point (graded)

In the space below, give a precise contract for `first-negative` , as described in the previous question. Be sure to format the contract as a comment, and to use the correct spelling, spacing, and punctuation.

```
;; first-negative: (listof Num) -> (anyof Num 'not-found)
```

Save Show answer

Submit You have used 1 of 5 attempts

Wrap-Up 05.8

Write a function `join` that consumes a non-empty list of strings `los`, together with an additional delimiter string `del`. The function produces a single string consisting of all the list elements joined together, separated by copies of `del`. For example, `(join (cons "A" (cons "B" (cons "C" empty))) "/")` would produce `"A/B/C"`.

1/1 points
Attempts: 1 / Unlimited

```
;; join: (listof Str) Str -> Str
;; Requires: los is non-empty
(define (join los del)
  (cond
    [(empty? los) ""]
    [(empty? (rest los)) (string-append (first los) "")]
    [else (string-append (first los) del
                         (join (rest los) del))]))
```

Submit Code Reset Code

Correct! 1/1 points

Wrap-Up 05.9

In a list of numbers, a given element is called a *peak* if it's strictly larger than all the numbers that come after it in the list. Write a function `peaks` that consumes a non-empty list of numbers and produces a sub-list consisting of all the peaks from the original list. For example, `(peaks (cons 1 (cons 6 (cons 4 (cons 5 empty))))) ` would produce `(cons 6 (cons 5 empty))`.

Your solution is likely to require a recursive helper function that determines if a number is larger than every element of a list, which you'll use at every step of the main function.

1/1 points
Attempts: 5 / Unlimited

```
;; peaks: (listof Num) -> (listof Num)
;; Requires: lon is non-empty
(define (peaks? s los)
  (cond
    [(empty? los) true]
    [(> s (first los)) (peaks? s (rest los))]
    [else false]))

(define (peaks lon)
  (cond
    [(empty? lon) empty]
    [(peaks? (first lon) (rest lon)) (cons (first lon) (peaks (rest lon)))]
  [else (peaks (rest lon))]))
```

Submit Code Reset Code

Correct! 1/1 points

Wrap-Up 05.10

Write a function `index` that consumes a value `val` of any type, and a list `lst`. The function produces a natural number giving the position of the first occurrence of `val` in `lst`, where counting starts at zero. For example, `(index 'a (cons 'c (cons 'a (cons 't empty))))` would produce `1`. You can assume that `val` appears at least once in `lst`.

1/1 points
Attempts: 2 / Unlimited

```
;; index: Any (listof Any) -> Nat
(define (index val lst)
  (cond
    [(equal? val (first lst)) 0]
    [else (add1 (index val (rest lst)))]))
```

Submit Code Reset Code

Correct! 1/1 points

Discussion

Hide Discussion

Topic: Module 05 / Wrap-Up Quiz

Add a Post

Show all posts | by recent activity

- ? 05.7
  Stuck on this question I don't know what else to add. ;; first-negative: (listof Num) -> (anyof Num Sym) 2
- ? 5.8 stuck
  Hello, I am stuck on this question because I can't find a ways to combine the list into a string. I tried to use append to combine the list but it isn't the right return. (define (join los de... 2
- ? Wrap-Up 05.9
  Why does (peaks (cons 1 (cons 6 (cons 4 (cons 5 empty))))) produce (cons 6 (cons 5 empty))? Does that mean (cons 1 (cons 7 (cons 3 empty))) would produce (cons 7 (cons 3 em... 4
- ? Wrap-Up 05.6
  I am stuck at this question. My code looks like this for now: (define (first-negative lon) (cond [(empty? lon) 'not-found] [(> 0 (first lon)) (first lon)] [(first-negative (rest lon))] [else 'n... 2

Previous Next