

CS 135 Winter 2020

Tutorial 04: Recursion on Lists and Integers

CS 135 Winter 2020

1

Announcements

- Assignment 4 is due on **Tuesday, Feb 11, at 9:00PM.**
- The times and location of office hours are posted on the "Office and Consulting Hours" page of the course website. Please email us at cs135@uwaterloo.ca to set up an appointment outside of these hours.

CS 135 Winter 2020

2

Goals of this tutorial:

By the end of this tutorial you should be able to...

- Utilize **list functions**.
- Understand and write **data definitions** for lists.
- Use the **template** for processing lists to write **recursive functions**.
- Write **recursive functions** on integers.

CS 135 Winter 2020

3

Review: Basic List Constructs 1/2

- `empty`: A value representing a list with 0 elements.
- `empty?`: Produces `true` if a given value is `empty` and `false` otherwise.
- `first`: Consumes a non-empty list and produces the first element.
- `rest`: Consumes a non-empty list and produces the same list without the first element.

Review: Basic List Constructs 2/2

- `cons`: Consumes a single element and a list, producing a new longer list.
- `cons?`: Produces `true` if a given value is a `cons` and `false` otherwise.
- `list?`: Produces `true` if a given value is a `list` and `false` otherwise.

Review: Substitution Rules

`(first (cons 'K empty))` \Rightarrow `K`

`(rest (cons 'K empty))` \Rightarrow `empty`

`(empty? empty)` \Rightarrow `true`

`(empty? (cons 'K empty))` \Rightarrow `false`

`(cons? (cons 'K empty))` \Rightarrow `true`

`(cons? 'K)` \Rightarrow `false`

`(cons? empty)` \Rightarrow `false`

`(list? empty)` \Rightarrow `true`

Clicker Question 1: List Template

Which of the options follows the `else` in `listof-X-template`?

```
;; listof-X-template: (listof X) → Any
```

```
(define (listof-X-template loX)
  (cond [(empty? loX) ...]
        [else (...)]))
```

- A (... (first loX) ... (rest loX) ...)
- B (... (first loX) ... (listof-X-template) ...)
- C (... (rest loX) ... (listof-X-template (first loX)) ...)
- D (... (first loX) ... (listof-X-template (rest loX)) ...)
- E (... (first loX) ... (listof-X-template (rest loX)) ... loX ...)

CS 135 Winter 2020

7

Problem 1: Sum Numbers

Based on the previous template, write a function called `sum-num` that consumes a list of numbers and produces the sum of those numbers.

Here are some examples:

```
(sum-num (cons 7 (cons 8 (cons 9 empty)))) ⇒ 24
```

```
(sum-num (cons 8 (cons 0 (cons 0 (cons 8 (cons -5 empty))))) ⇒ 11
```

CS 135 Winter 2020

8

Problem 1: Design Recipe

```
;; (sum-num lon) Produces the sum of all numbers in lon.
```

```
;; sum-num: (listof Num) → Num
```

```
;; Examples:
```

```
(check-expect (sum-num (cons 5 (cons .5 empty))) 5.5)
```

```
(check-expect (sum-num empty) 0)
```

```
(define (sum-num lon) ...)
```

```
;; Tests:
```

```
(check-expect (sum-num (cons 0 empty)) 0)
```

```
(check-expect (sum-num (cons 5 (cons -5 empty))) 0)
```

CS 135 Winter 2020

9

Clicker Question: Debugging Recursive Functions

How many errors are there in the following function?

;; Add-downto-3 produces the sum of all integers between n and 3 (inclusive)

;; Add-downto-3: Nat → Num

```
(define (Add-downto-3 n)
  (cond
    [(= n 3) 0]
    [else (+ n (Add-downto-3 (sub1 n)))]))
```

- A It is a perfect function!
- B 2
- C 4
- D 5
- E Too many to count.

CS 135 Winter 2020

10

Problem 2: Add-between

Based on the previous question, write a function called `Add-between` that consumes two integers and produces the sum of all integers between them (inclusive).

Here are some examples:

`(Add-between 3 5) ⇒ 12`

`(Add-between 0 3) ⇒ 6`

CS 135 Winter 2020

11

Problem 2: Design Recipe

;; (Add-between a b) Produces the sum of all integers between a and b

;; Add-between: Int Int → Int

;; Examples:

`(check-expect (Add-between 3 5) 12)`

`(check-expect (Add-between 0 3) 6)`

```
(define (Add-between a b) ...)
```

;; Tests:

`(check-expect (Add-between 3 3) 3)`

`(check-expect (Add-between 4 3) 7)`

CS 135 Winter 2020

12

Problem 2: Another Solution

```
(define (Add-between-acc a b sofar)
  (cond
    [(= a b) (+ a sofar)]
    [(> a b) (Add-between-acc (sub1 a) b (+ a sofar))]
    [else (Add-between a (sub1 b) (+ b sofar))]))

(define (Add-between a b)
  (Add-between-acc a b 0))
```

CS 135 Winter 2020

13

Extra Practice: strings-equal?

Write a function called `strings-equal?` that consumes a list of strings and produces `true` if all of the strings in the list are equal and `false` otherwise.

Here are some examples:

```
(strings-equal? empty) ⇒ true
(strings-equal? (cons "cs" (cons "cs" empty))) ⇒ true
(strings-equal? (cons "cs" (cons "se" (cons "cs" empty)))) ⇒ false
```

Hint: The template only includes one base case. However, some functions need multiple base cases.

CS 135 Winter 2020

14

Extra Practice: Diagonal

Define a function that consumes a Nat `length` and produces a square table of that length where all entries on the diagonal (top-left to bottom-right) being 1 and the rest are 0.

```
(draw-diagonal 0) ⇒ empty
(draw-diagonal 4) ⇒ (list (list 1 0 0 0)
                          (list 0 1 0 0)
                          (list 0 0 1 0)
                          (list 0 0 0 1))
```

CS 135 Winter 2020

15