605.629: Programming Languages
Assignment 7
Sabbir Ahmed

October 23, 2021

1. [20 pts] Convert the following Scheme code to tail recursive:

```scheme
(define length
  (lambda (lat)
    (cond
      ((null? lat) 0)
      (else (+ 1 (length (cdr lat))))
)))

(length '(a b c d e f)) => 6
```

Answer

```scheme
(define lengthr                         ; convert core function into helper
  (lambda (lat i)                       ; add an accumulator as an argument
    (cond
      ((null? lat) i)                   ; if empty list, return the accumulator
      (else
        (lengthr (cdr lat) (+ 1 i)))    ; move incrementing into argument
)))

(define length                          ; wrap helper function to provide default argument
  (lambda (lat)
    (lengthr lat 0)
))

(length '(a b c d e f)) => 6
```

---

2. [40 pts] Write a Scheme function `(mean lst)` to compute the mean value of a list of integers. Make sure the function traverses the list once and computes both the sum and the length in order to return the mean (i.e. sum/len).

Answer

```scheme
(define sum
  (lambda (lat)
    (cond
      ((null? lat) 0)
      (else (+ (car lat) (sum (cdr lat))))
)))

(define mean
  (lambda (lat)
    (/ (sum lat) (length lat))
))
```

---

3. [40 pts] Recall that lazy variant or call-by-need, is an evaluation strategy which delays the evaluation of an expression until its value is needed and which also avoids repeated evaluations. Given,

```
(define foo
  (lambda (x y z)
    (if x (1 + y) (1 - z))
))
```

a. What happens if we enter `(foo #t 7 (quotient 1 0))`

i.) in Scheme?

Answer

A divide-by-zero error is thrown when attempting to calculate `(quotient 1 0)` before being passed as an argument to `foo`.

---

ii.) in a lazy variant of Scheme?

Answer

A lazy variant of Scheme will output `8` since its first argument `x = #t` branches to `(+ 1 y)` where `y = 7`. Since the function does not need to compute `z`, `(quotient 1 0)` is never evaluated.

---

b. What happens if we enter

```
(foo #t 2 (letrec ([loopy (lambda (x)
  (if (zero? x)
    0
    (loopy (1 + x))))
]) (loopy 1)))
```

i.) in Scheme?

Answer

`(loopy 1)` is evaluated and throws the interpreter in an endless loop. Its argument is checked in the conditional `(zero? x)` where `x = 1`, evaluates to false and branches to the recursive call to the function with `x` forever incrementing.

---

ii.) in a lazy variant of Scheme?

Answer

A lazy variant of Scheme will output `3` since its first argument `x = #t` branches to `(+ 1 y)` where `y = 2`. Since the function does not need to compute `z`, `loopy` is never evaluated.