CIS 352 **Name:** _____
Spring 2026 **Midterm 2 PRACTICE**
Default Time Limit: 80 minutes

- You may not use anything except for a single **two-sided** *handwritten (by you)* note sheet of US letter size (8.5 x 11 inches) paper. You may not collaborate with anyone, Google for answers, use ChatGPT, etc...
- If you notice what you think could be a mistake, please document your thinking for partial credit.
- We will not penalize small syntactic issues (judged by instructor); you should document all of your relevant thoughts for potential partial credit—we will grade as consistently as possible, and will attempt to award partial credit when possible.

Problem	Points	Score
1	3	
2	4	
3	3	
4	4	
5	3	
6	2	
7	4	
8	2	
Total:	25	

1. (3 points) Define the function g by the following identity: $((g\ f)\ x) = (f\ (f\ x))$
   Write a definition of `g` as a Racket lambda (you can use either the word lambda or λ):

```
(define g (lambda (f) ...))
```

2. (4 points) Consider the λ-calculus term $\Big(\big(\lambda(x)\ (\lambda(z)\ k)\big)\ \ \big((\lambda(x)(x\ x))\ (\lambda(l)(l\ l))\big)\Big)$. The term has two β-redexes: perform both of them, and show the result of each of the two β reductions.

3. (3 points) The following is a direct-style (non-tail-recursive) Racket function:

```
(define (sum-em l f)
  (if (empty? l)
      0
      (+ (sum-em (rest l) f)
         (if (f (first l)) (first l) 0))))
```

   Rewrite the function so that it is *tail-recursive*. (Essential approach: create a helper function that takes an extra acc argument. Make sure the helper function calls itself only in tail position!)

4. (4 points) Consider a λ-calculus term e. Assume that $e \overset{\eta,\alpha,\beta}{\Longrightarrow}{}^{*}\ v$ (i.e., "e reduces in multiple steps of η, α, or β to v) where v is in β-normal form, and also that $e \overset{\eta,\alpha,\beta}{\Longrightarrow}{}^{*}\ v'$ where v' is also in β-normal form. **Please answer both (a)** are v and v' identical terms?; **(b)** are they α-equivalent?; and, **(c)** why or why not?

5. (3 points) Consider both foldl and foldr:

```
(define (foldr f i l)
  (match l
    ['() i]
    [`(,hd ,tl ...) (f hd (foldr f i tl))]))
(define (foldl f acc l)
  (match l
    ['() acc]
    [`(,hd ,tl ...) (foldl f (f hd acc) tl)]))
```

Consider that f is a function which is commutative and also associative, i.e., $(f\ x\ y) = (f\ y\ x)$ and $(f\ (f\ x\ y)\ z) = (f\ x\ (f\ y\ z))$. Does (foldr f i l) = (foldl f i l)? You must briefly **explain** your answer (correct answer with <u>no</u> explanation will get **1/3** points).

6. (2 points) Assume e is a term in the λ-calculus. Using a call-by-name evaluation strategy, it reduces to β-normal form v. Consider every other reduction sequence starting from $e \Longrightarrow e'$—do all of them terminate? Why or why not?

7. (4 points) Consider the following system, specified via natural deduction:

$$\frac{}{\#t \Downarrow \#t}\ \text{E-True} \qquad \frac{}{\#f \Downarrow \#f}\ \text{E-False} \qquad \frac{e \Downarrow \#t}{(\mathsf{not}\ e) \Downarrow \#f}\ \text{E-NotTrue}$$

$$\frac{e \Downarrow \#f}{(\mathsf{not}\ e) \Downarrow \#t}\ \text{E-NotFalse} \qquad \frac{e_1 \Downarrow \#t \qquad e_2 \Downarrow v}{(\mathsf{if}\ e_1\ e_2\ e_3) \Downarrow v}\ \text{E-IfTrue} \qquad \frac{e_1 \Downarrow \#f \qquad e_3 \Downarrow v}{(\mathsf{if}\ e_1\ e_2\ e_3) \Downarrow v}\ \text{E-IfFalse}$$

   Give a derivation proving that $(\mathsf{if}\ (\mathsf{not}\ \#t)\ \#t\ \#f) \Downarrow \#f$.

8. (2 points) In the context of functional programming (in Racket and similar languages), explain (1-2 sentences) the difference between a *lambda* and a *closure*.