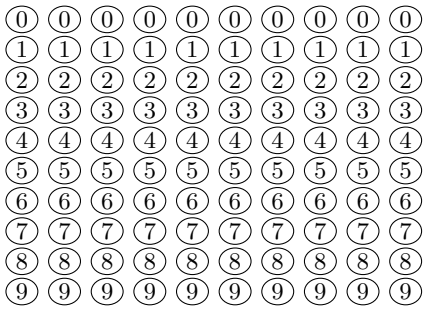
CSC 324H5 F 2019 Midterm
Duration — 50 minutes
Aids allowed: none

Student Number:



UTORid: _____

Last Name: _____ **First Name:** _____

Lecture Section: L0102 Test Version: B Instructor: Lisa Zhang

---

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above, and read the instructions below.)
Good Luck!

This test consists of 5 questions on 8 pages (including this page). *When you receive the signal to start, please make sure that your copy is complete.*
Comments are not required except where indicated, although they may help us mark your answers. They may also get you part marks if you can't figure out how to write the code.
If you use any space for rough work, indicate clearly what you want marked.

1: _____/ 7

2: _____/ 3

3: _____/ 5

4: _____/ 5

5: _____/ 5

TOTAL: _____/25

Total Pages = 8

Question 1. [7 MARKS]

Part (a) [2 MARKS]

In Racket, can `if` expressions of the form `(if <cond> <then> <else>)` be implemented as a function? Why or why not?

Part (b) [3 MARKS]

What do the following Racket expressions evaluate to? If there is an error, explain why.

```
> (define (f x) (lambda (x) x))
> (f 1)
```

```
> (f 1 2)
```

```
> ((f 1) 2)
```

Part (c) [2 MARKS]

What do the following Haskell expressions evaluate to? If there is an error, explain why.

```
Prelude> g x y = x
Prelude> g 2
```

```
Prelude> g 4 (1 / 0)
```

Question 2. [3 MARKS]

Part (a) [2 MARKS]

Consider the calculator grammar from Exercise 4:

```
<expr> ::= NUM                  ;; integer in base 10
         | ID                   ;; variable names (excluding +, -, *, /, =, >, <)
         | (<op> <expr> <expr>)
         | (if (<comp> <expr> <expr>) <expr> <expr>)
         | (let* ((ID <expr>) ...) <expr>)
         | (lambda (ID ...) <expr>)
         | (<expr> <expr> ...)
<comp> ::= = | > | <
<op>   ::= + | - | * | /
```

Cross out any of the four expressions below that are **not syntactically valid in this grammar**.

- `(lambda (x) (= x (+ 3 4)))`
- `()`
- `(if (= (lambda (x) x) (lambda (y) y)) 3 4)`
- `(3 4 5)`

Part (b) [1 MARK]

What is a closure?

Question 3. [5 MARKS]

Part (a) [2 MARKS]

Consider the following macro:

```
(define-syntax my-macro
  (syntax-rules ()
    [(my-macro (<x> (<y>) ...)) (list 'a <x>)]
    [(my-macro (<x>))           (list 'b <x>)]
    [(my-macro (<x> <y> ...))   (list 'c <y> ...)]))
```

Perform macro expansion on the following two expressions. Write "ERROR" without further explanation if there is an error.

```
(my-macro (1))
```

```
(my-macro (1 (2) 3))
```

Part (b) [3 MARKS]

Create a class `Pet` with the attribute `name` and `age`, and a method `birthday` that returns a new `Pet` with the same name, and with the `age` incremented by one.

Question 4. [5 MARKS]

Recall that we can use `cond` in Racket to write conditional statements. However, any code written in terms of `cond` can be rewritten using nested calls to `if`.

```
(define (f x)
  (cond [(< x 3) (+ x 3)]
        [(> x 5) (* x 2)]
        [else    (+ x 1)]))
```

```
(define (f x)
  (if (< x 3)
      (+ x 3)
      (if (> x 5)
          (* x 2)
          (+ x 1))))
```

Complete the following implementation of a macro `my-cond`. The arguments of `my-cond` follow the Racket syntax for `cond`; the macro rewrites the expression in terms of equivalent calls to `if`. You may assume that the final condition of `cond` will be an `else`.

```
(define-syntax my-cond
  (syntax-rules (else)
   [(my-cond [else <expr>])
     [                                    ]

    [(my-cond [<cond> <expr>] <rest> ...)
     (if [                                ]
         [                                ]
         ( [                              ] [                              ] ))]))
```

Question 5. [5 MARKS]

Write a function `merge` in Racket that merges two sorted lists. For example:

```
> (merge '(1 4 5) '(2 4))
'(1 2 4 4 5)
> (merge '(1 4 5) '())
'(1 4 5)
> (merge '() '(5))
'(5)
```

For full marks, use tail recursion. A non-tail recursive solution can earn up to 3 points. You may write as many helper functions as you need, and use any of the list functions in the aid sheet.

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]