CSC 324H1 S February Midterm 2016
Duration — 50 minutes

**Student Number:** _ _ _ _ _ _ _ _ _

**Family Name:** _____ **Given Name:** _____

---

*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 quiz consists of 4 questions on 5 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.* If you do any rough work, indicate clearly what you want marked.

# 1: _____/10

# 2: _____/ 5

# 3: _____/10

# 4: _____/10

TOTAL: _____/35

---

## Question 1. [10 MARKS]

For this question: use higher-order-functions 'map', 'apply', and/or 'filter' wherever appropriate.

### Part (a) [6 MARKS]

Write 'pull-back' that takes a unary function 'f', a domain list 'D', and a value 'y', and produces a list of each value in 'D' such that 'f' at the value is equal to 'y'.

```
(check-expect (pull-back sqr '(-3 2 1 -2) 4)
              '(2 -2))
(check-expect (pull-back sqr '(-3 2 1 -2) 1)
              '(1))
```

### Part (b) [4 MARKS]

Write 'pull-backs' that is like 'pull-back' but it takes a list of y values 'ys' and produces a list of each value in 'D' such that 'f' at the value is equal to one of the values in 'ys'. You may use 'pull-back'.

```
(check-expect (pull-backs sqr '(-3 2 1 -2) '(1 4))
              '(1 2 -2))
```

## Question 2. [5 MARKS]

Assume that 'and' is not in the racket library, but 'or' and 'not' are.
Write a macro to define 'and', that works with any number of boolean expressions.
Hint: recall DeMorgan's Law: $\neg(\neg P \vee \neg Q) \equiv P \wedge Q$.

## Question 3. [10 MARKS]

Show the tree of environments and closures for the following code, and the result value of the final expression:

```
(define (f x)
  (set! x (+ x 1))
  (lambda () x))
(define x (f 1))
(define y (f 20))
(y)
```

## Question 4. [10 MARKS]

Consider the following runtime data language of propositional formulas.

A symbol is an atomic propositional formula. If P and Q are propositional formulas, then so are the lists:

- (P ∧ Q)
- (P ∨ Q)
- (¬ P)

Write a function 'move' that takes a propositional formula and moves the negations inward, producing a logically equivalent formula that has negation only in the form (¬ A) where A is an atomic propositional formula.
Recall:

- $(\neg(\neg P)) \equiv P$
- $(\neg(P \wedge Q)) \equiv ((\neg P) \vee (\neg Q))$
- $(\neg(P \vee Q)) \equiv ((\neg P) \wedge (\neg Q))$

```
(check-expect (move '(a ∧ (¬ (b ∨ (¬ c))))) '(a ∧ ((¬ b) ∧ c)))
```

Use 'match'.
Use quasi-quotation in the patterns and the result expressions where worthwhile.

Hint: include a clause or clauses for non-negated conjunction and disjunction.

Total Marks = 35