PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

APRIL 2016 EXAMINATIONS

CSC 104 H1S
Instructor(s): G. Baumgartner

Duration — 3 hours

No Aids Allowed

Student Number:	
Last (Family) Name(s):	
First (Given) Name(s):	

Do ***not*** *turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

Marking Guide

# 1:	______/ 7
# 2:	______/ 9
# 3:	______/ 8
# 4:	______/ 2
# 5:	______/25
# 6:	______/11
# 7:	______/ 4
# 8:	______/10
# 9:	______/ 2
TOTAL:	______/78

This Final Examination paper consists of 9 questions on 16 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the paper is complete and fill in your name and student number above.*

Answer each question directly on this exam paper, in the space provided. If you need more space for one of your solutions you may also use a "blank" page at the end of the paper (in which case make sure to mention that where the question is asked).

You will earn 20% for any question you leave blank or you write "I cannot answer this question".

Good Luck!

Reminders

The Intermediate Steps for `map`, `apply`, and `repeated`:

```
(map f (list    a     b     c  ...))

       (list (f a) (f b) (f c) ...)


(map f (list    a       b       c    ...)
       (list    x       y       z    ...))

       (list (f a x) (f b y) (f c z) ...)


(apply f (list a b c ...))

         (f    a b c ...)


; repeated :  function any number  ->  list
(define (repeated f a n)
  (cond [(= n 1) (list a)]
        [else (list* a (repeated f (f a) (- n 1)))]))

(repeated f a n)

(list a (f a) (f (f a)) ...) ; with n elements.
```

The function `range`:

```
(check-expect (range 12 34 5) (list 12 17 22 27 32))
```

The function `list-ref`:

```
; list-ref : list number -> any

(check-expect (list-ref (range 12 34 5) 3)
              27)
```

Question 1. [7 marks]

Part (a) [4 marks]

Multiply the numbers whose binary representations are 1011 and 1101.
Use the multiplication algorithm **in binary notation**, showing your work.

Part (b) [3 marks]

The number 104104 has the binary reprsentation:

- 11001011010101000

Use that fact to determine the decimal representations of:

- 11001011010101001
- 110010110101010000

Give/show a brief explanation of your reasoning.

Question 2. [9 MARKS]

Consider this function T:

```
(define (T n)
  (cond [(zero? n) (rotate-ccw (triangle 10 "outline" "black"))]
        [else (beside (T (- n 1))
                      (flip-horizontal (above (T (- n 1))
                                              (T (- n 1)))))]))
```

Part (a) [3 MARKS] Draw the values of `(T 0)` and `(T 1)`:

Part (b) [2 MARKS]

Complete this "design" `check-expect`, showing the intermediate step that uses `(T 1)`:

```
(check-expect (T 2)
```

Part (c) [4 MARKS] Draw the values of `(T 2)` and `(T 3)`:

Question 3. [8 marks]

Part (a) [4 marks]

Recall from Project I the function `random-element` that takes a non-empty list and produces a random element from the list. Define function `random-elememt`:

```
; random-element : list -> any
```

Part (b) [4 marks]

Define a function `p?` that takes a list of two numbers and two numbers `x` and `y`.
It produces `#true` if the numbers in the list are between 0 and 104 inclusive, and at least one of the numbers `x` or `y` is negative, otherwise it produces `#false`:

```
; p? : list-of-two-numbers number number -> boolean
(define (p? a-list x y)
```

Question 4. [2 marks]

Determine the right-most decimal digit of the value of the following expression, briefly showing how you calculated that digit:

```
(* 1234567890123456789123 (sqr 987654321098765432))
```

In other words, what is the value of:

```
(remainder (* 123456789012345678912 (sqr 98765432109876543))
           10)
```

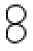
Question 5. [25 MARKS]

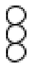
Part (a) [3 MARKS]

Here are four `check-expect`s for a function `c0`:

```
(check-expect (c0 1) (circle 10 "outline" "black"))
```

`(check-expect (c0 1)` `)`

`(check-expect (c0 2)`  `)`

`(check-expect (c0 3)`  `)`

Complete the following two `check-expects` **without** drawing any images manually by hand.
Use `(c0 1)` in the first `check-expect`.
Use `(c0 2)` in the second `check-expect`.

```
(check-expect (c0 2)



)
(check-expect (c0 3)



)
```

Part (b) [6 MARKS]

Fill in the contract for the function `c0`, and then define the function according to your `check-expect`s:

```
; c0 :                    ->
```

Part (c) [1 MARK]

Consider this function:

```
; S : any -> image
(define (S unused)
  (circle 10 "solid" "black"))
```

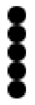
Show the value of the following expression:

```
(S "hello")
```

Part (d) [2 marks]
Write an expression that uses `S` and `range` to produce this list of five circles: `(list ● ● ● ● ●)` .

Part (e) [2 marks]
Assume `circles` is defined by:

```
(define circles (list ● ● ● ● ●))
```

Write an expression that uses `circles` to produce:  .

Part (f) [1 mark]
Consider the following "partial design" `check-expect` for a function `cS`:

```
(check-expect (cS 5)
              (above (S 0) (S 1) (S 2) (S 3) (S 4)))
```

Complete the following `check-expect` as a "full design" `check-expect`, based on the approach in the "partial design" and Parts (c-e).

```
(check-expect (cS 5)
```

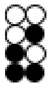
Part (g) [2 marks]
Define the function `cS` according to your "full design" `check-expect`:

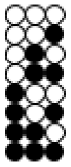
Part (h) [4 MARKS]

Consider the following `check-expects` describing a function `C`:

```
(check-expect (C 1) (circle 0 "solid" "transparent"))
```

`(check-expect (C 2)` `)`

`(check-expect (C 4)`  `)`

`(check-expect (C 8)`  `)`

Complete the following as a "design" `check-expect` by using `(C 4)`, and the functions `c0` and `cS`:

```
(check-expect (C 8)
```

Part (i) [4 MARKS]

Define the function `C`.
You may assume the input `n` is a power of 2.

```
(define (C n)
```

Question 6. [11 marks]

Consider this function `f`:

```
(define (f LLL)
  (cond [(list? LLL) (+ (length LLL)
                        (apply + (map f LLL)))]
        [else 1]))
```

Part (a) [3 marks] Show the values of the following expressions:

```
(f "A")
```

```
(f (list "B" "C" "D"))
```

Part (b) [5 marks]

Show some intermediate steps and final result value for the following expression.
You may use the results from Part (a) without showing their steps.
But show (at least) the other steps where `apply` or `map` are performed: include the expression containing `apply` or `map`, followed by the intermediate step expression (as shown in the the "Reminders" on the second page of the exam), followed by the step after that.

```
(f (list "A" (list "B" "C" "D")))
```

Part (c) [3 MARKS]

Show the final result value for the following expression.
Inlcude a brief explanation, or illustrative intermediate step(s):

```
(f (list (list "A"
               (list "B" "C" "D"))
         "E"))
```

Question 7. [4 MARKS]

Show the intermediate step(s) and results value for each of the following expressions:

Part (a) [2 MARKS]

```
(apply append (list (list (list 1 2) (list 3 4))
                    (list (list 5 6) (list 7 8))))
```

Part (b) [2 MARKS]

```
(map append
     (list (list 1 2) (list 3 4))
     (list (list 5 6) (list 7 8)))
```

Question 8. [10 marks]

Consider the following definitions:

```
; R : number -> image
(define (R n)
  (rectangle 10 (* 10 n) "outline" "black"))

; draw : list -> image
(define (draw p)
  (beside (R (first p)) (R (second p))))

; update : list -> list
(define (update p)
  (cond [(= (second p) 2) (list (+ 1 (first p)) 1)]
        [else (list (first p) 2)]))
```

Part (a) [4 marks]

Show the values of each of the following expressions:

```
(R 1)
```

```
(draw (list 1 1))
```

```
(update (list 1 1))
```

Part (b) [4 marks]

For the following expression, show (at least) the intermediate step for `repeated`, and the final result value:

```
(repeated update (list 1 1) 5)
```

Part (c) [2 MARKS]

For the following expression, show (at least) the intermediate step for `map`, and the final result value:

```
(map draw (repeated update (list 1 1) 5))
```

Question 9. [2 MARKS]

Consider the following function to compute powers of 2:

```
; expt-2 : number -> number
(define (expt-2 n)
  (cond [(zero? n) 1]
        [else (+ (expt-2 (- n 1))
                 (expt-2 (- n 1)))]))
```

Explain the problem that occurs when someone tries `(expt-2 100)` to calculate 2^{100}.

Use the space on this "blank" page for scratch work, or for any answer that did not fit elsewhere.
Clearly label each such answer with the appropriate question and part number.

Use the space on this "blank" page for scratch work, or for any answer that did not fit elsewhere.
Clearly label each such answer with the appropriate question and part number.

Please write nothing on this page.