

91.301 Quiz 1 review

Please write your name on this exam sheet and hand it in along with any blue books that you may use. This exam is open book: you may use your textbook, course notes, homework problems and solutions to work this exam – if you have brought them. You may not use any electronic device.

Problem 1. [15] Basic Scheme expressions.

Assume that the following are typed in the DrScheme interaction window in the order in which they appear. After each write down what you expect the DrScheme interpreter to print. Do not worry about the prompt, or the details of error messages or procedure representations.

```
(define (y) 2)
```

y

```
(y)
```

```
(define x 5)
```

x

```
(x)
```

```
(x y)
```

```
(y x)
```

```
(let ((y 4)
      (z 9))
      (+ x y x))
```

y

z

```
(let ((+ *)
      (* +))
      (+ (* 2 3) (* 2 3)))
```

```
(let ((+ *))
      (let ((* +))
        (+ (* 2 3) (* 2 3))))
```

Answer: nothing is printed for the definition

Answer: <procedure>

Answer: 2

Answer: nothing is printed for the definition

Answer: 5

Answer: error: x is not a procedure

Answer: error: x is not a procedure

Answer: error: procedure y given 1 argument, takes 0

Answer: 18

Answer: <procedure>

Answer: error: unbound variable z

Answer: 25

Answer: 36

Problem 2. [35] Substitution model, higher-order functions.

```
(define (compose f g)
  (lambda (x) (g (f x))))
```

Assume that `dec` is a primitive procedure that subtracts 1 from its argument. Show all steps in the substitution model of evaluating

```
((compose
  (compose dec dec)
  dec)
  5)
```

Answer:

Order is irrelevant, but must evaluate operator and operands before applying.

```
((compose (compose dec dec) dec) {5})
((compose ({proc (f g) (lambda (x) (g (f x))))} dec dec) dec) {5})
((compose ({proc (f g) (lambda (x) (g (f x))))} {dec} dec) dec) {5})
((compose [{proc (f g) (lambda (x) (g (f x))))} {dec} {dec}] dec) {5})
((compose (lambda (x) ({dec} ({dec} x))) dec) {5})
((compose {proc (x) ({dec} ({dec} x))} dec) {5})
((compose {proc (x) ({dec} ({dec} x))} {dec}) {5})
([{proc (f g) (lambda (x) (g (f x))))} {proc (x) ({dec} ({dec} x))} {dec}] {5})
((lambda (x) ({dec} ({proc (x) ({dec} ({dec} x))} x))) {5})
[{proc (x) ({dec} ({proc (x) ({dec} ({dec} x))} x))} {5}]
{dec} [{proc (x) ({dec} ({dec} x))} {5}]
{dec} {dec} [{dec} {5}]]
{dec} [{dec} {4}]]
[{dec} {3}]
{2}
```

Problem 3. [25] Recursive procedures.

Part a. [10]

Write a routine (`int-log x y`) which returns the number of times that `x` needs to be divided by `y` to return a number less than or equal to 1.

Answer:

```
(define (int-log x y)
  (if (<= x 1)
      0
      (+ 1 (int-log (/ x y) y))))
```

Part b. [5]

Does your procedure generate an iterative process or a recursive process?

Answer: recursive

Part c. [10]

If your procedure generated an iterative process, write another version that generates a recursive process. If your procedure generated a recursive process, write another version that generates an iterative process.

Answer:

```
(define (int-log x y)
  (define (iter x result)
    (if (<= x 1)
        result
        (iter (/ x y) (+ 1 result))))
  (iter x 0))
```

Problem 4. [25] Orders of growth.

Assume we have defined `sum` and `square` as follows:

```
(define sum
  (lambda (term a next b)
    (if (> a b)
        0
        (+ (term a)
            (sum term (next a) next b)))))
```

```
(define square
  (lambda (x) (* x x)))
```

```
(define id
  (lambda (x) x))
```

What is the order of growth in time for the processes generated by the following procedures? Your answer for each should be one of $\Theta(1)$, $\Theta(\log n)$, $\Theta(n)$, $\Theta(n^2)$, or $\Theta(2^n)$.

```
(define (s1 n)
  (sum id
    1
    (lambda (i) (* i 2))
    n))
```

```
(define (s2 n)
  (sum (lambda (i) (sum id 1 inc i))
        1
        inc
        n))
```

```
(define (s3 n)
  (sum (lambda (i) (/ (* i (+ i 1)) 2))
        1
        inc
        n))
```

```
(define (sum-of-squares x y)
  (+ (square x)
      (square y)))
```

```
(define (number-of-bits-in n)
  (if (< n 2)
      1
      (+ 1 (number-of-bits-in (/ n 2)))))
```

Answer: $\Theta(\log n)$, $\Theta(n^2)$, $\Theta(n)$, $\Theta(1)$, $\Theta(\log n)$.