Problem #1 (7 points):
What will Scheme print in response to the following expression? If an expression produces an error message or runs forever without producing a result, you may just say "error"; you don't have to provide the exact text of the message. If the value of an expression is a procedure, just say "procedure"; you don't have to show the form in which Scheme prints procedures. Assume that no global variables have been defined before entering these expressions, except where noted.

(se '(+ 2 3) (+ 2 3))
((lambda (x y z) (+ x 5)) 6 7)
; from ex. 1.32, p. 61
(accumulate se 0 (lambda (x) x) 3 (lambda (x) (+ x 1)) 5)
((if 3 - *) 23 2)
(a b c)
(let ((a 5) (b (+ a 3))) (* a a))
((lambda (f) (f f)) (lambda (f) f))

Problem #2 (2 points):
True or false?
A theta( nlog(n)) algorithm is, for all large enough n, slower than a theta ( n^2) one. _____
For small size inputs the theta order of an algorithm helps predict running time. _____
Function f below defines a linear iterative process:
(define (f a b c)
  (if (> a b)
      c
      (f (+ a 1) (- b 1) (+ c 1)))))
Function g below defines a linear iterative process:
(define (g a b c)
  (if (> a b)
      c
      (+ c (g (+ a 1) (- b 1) (+ c 1)))))

Problem #3 (10 points):
Write a function `stutter` that takes a word `w` and a number `n` and produces a function. This function takes a sentence `s` and for EVERY recurrence of the word `w` it reproduces it `n` times.

For example

```
(define porky (stutter 'th 3))
(porky '(th thats all folks))
```
evaluates to `(th th th thats all folks)`. You may need to define a helper function, too.

**Problem #4 (8 points):**

```
(define (ss k)
  (define (tt k r)
    (if (empty? k)
      r
      (tt (bf k) (se (first k) r))))
  (tt k '(d)))
```

Write out (or "trace") the succession of calls to `ss` and `tt`, and their return values as Scheme evaluates the expression `(ss '(a b c))`.

Is the process traced out with `tt` linear iterative?

**Problem #5 (12 points):**

Sometimes you want to reduce a collection of elements by operating on them in pairs, starting from the right, and given an end-value when there is only one element left. For example `(reduce + '(2 5 6) 0)` is meant to compute `(+ 2 (reduce '(5 6) 0))` which is, in turn, equivalent to `(+ 2 (+ 5 (reduce '(6) 0)))` which is `(+ 2 (+ 5 (+ 6 (reduce '() 0))))` which is `(+ 2 (+ 5 (+ 6 0)))` or 13.

You may need a few extra "helper" procedures to complete these programs. Use the reverse of this page if you need more space.

A. Define the procedure `(reduce f s e)` illustrated above that takes as its argument another procedure `f`, a sentence `s`, and an end-value `e`. Procedure `f` should take two arguments.

B. Use `reduce` to reverse the order of words in a sentence. That is, define a procedure `reverse-by-reduce` that given `(hello good bye)` returns `(bye good hello)`.

C. Use `reduce` to find a word with the largest number of letters in a given sentence. That is, define a procedure `longest` that given `(two three five)` returns `three`.

D. Use `reduce` to find the minimum number in a given sentence `r`. That is, define a procedure `minimum` that given `(0 -500 30)` returns `-500`. If `r` is empty, return the word `error`.
If you have any questions about these online exams please contact examfile@hkn.eecs.berkeley.edu.