Problem 1A.

```
(value-of
      <<let a = 10 in
            let f = proc (func) (func a) in
                  let a = 5 in
                        let g = proc (x) -(x, a) in (f g)>>
      ρ)

Let ρ1 = [a=[10]]ρ

= (value-of
      <<let f = proc (func) (func a) in
            let a = 5 in
                  let g = proc (x) -(x, a) in (f g)>>
      ρ1)

Let ρ2 = [f=(proc-val (procedure func <<(func a)>> ρ1)]ρ1

= (value-of <<let a = 5 in let g = proc (x) -(x, a) in (f g)>> ρ2)

Let ρ3 = [a=[5]]ρ2

= (value-of <<let g = proc (x) -(x, a) in (f g)>> ρ3)

Let ρ4 = [g=(proc-val (procedure x <<-(x, a)>> ρ3))]ρ3

= (value-of <<(f g)>> ρ4)

= (apply-procedure
      (procedure func <<(func a)>> ρ1)
      (procedure x <<-(x, a)>> ρ3))

= (value-of
      <<(func a)>>
      [func=(proc-val (procedure x <<-(x, a)>> ρ3))]ρ1)

= (apply-procedure (procedure x <<-(x, a)>> ρ3) [5])

= (value-of <<-(x, a)>> [x=[5]]ρ3)

= (- 5 5)

= 0
```

Problem 2A

Check for comments in “tests.rkt” for explanations.

Problem 2B

Procedure arguments are now a list (including the empty list).

Modified `proc-exp` and `call-exp` grammar in “lang.rkt”
Modified `proc-exp` and `call-exp` evaluation in “interp.rkt”
Modified `apply-procedure` in “interp.rkt”
Modified `procedure` datatype in “data-structures.rkt”