

# PhD Qualifying Examination

## Programming Languages

### Spring 2006

**Open book, open notes, closed laptop, no collaboration**

1. What is the value of this expression assuming Scheme uses static scoping? What is the value assuming that Scheme uses dynamic scoping?

```
(let ((x 3))
  (let ((f (lambda (y) (+ x y))))
    (let ((g (lambda (x) (f 10))))
      (g 100))))
```

2. What is displayed by this program, assuming the language uses call-by-value? What is displayed, assuming the language uses call-by-name?

```
(let ((x 3))
  (let ((f (lambda (y) (let ((x 10)) (begin (display "Result: ") (display (+ x x))))))
    (f (begin (display "Actual: ") x))))
```

3. Here is a little interpreter. Fill in the missing parts assuming the language is (a) call-by-*value* statically scoped (b) call-by-*value* *dynamically* scoped (c) call-by-*name* statically scoped (d) call-by-*name* *dynamically* scoped.

```
(define-datatype exp exp?
  (const-exp (Num number?))
  (var-exp (Sym symbol?))
  (plus-exp (Expl exp?) (Expr exp?))
  (app-exp (Rator exp?) (Rands (list-of exp?)))
  (proc-exp (Params (list-of symbol?)) (Body exp?)))
```

```
(define-datatype value value?
  (num (N number?))
  (closure ....) // A
  (thunk ....)) // B: This is not required for all parts
```

```
(define (value->num v) (cases v value (num (N) N)))
```

```
(define (eval-exp E Env)
  (cases E exp
    (const-exp (Num) (num Num))
    (var-exp (Sym) (let ((v (lookup-env Sym Env)))) ...)
    (plus-exp (expl expr) (+ (eval (value->num expl) Env)
                              (eval (value->num expr) Env)))
    (proc-exp (Params Body) (closure ...) // C
    (app-exp (Rator Rands) (apply-proc (eval-exp Rator Env)
                                         Rands Env))))
```

```
(define (apply-proc Proc Actuals Env)
  (cases Proc value
    (closure (...) ...))) // D
```