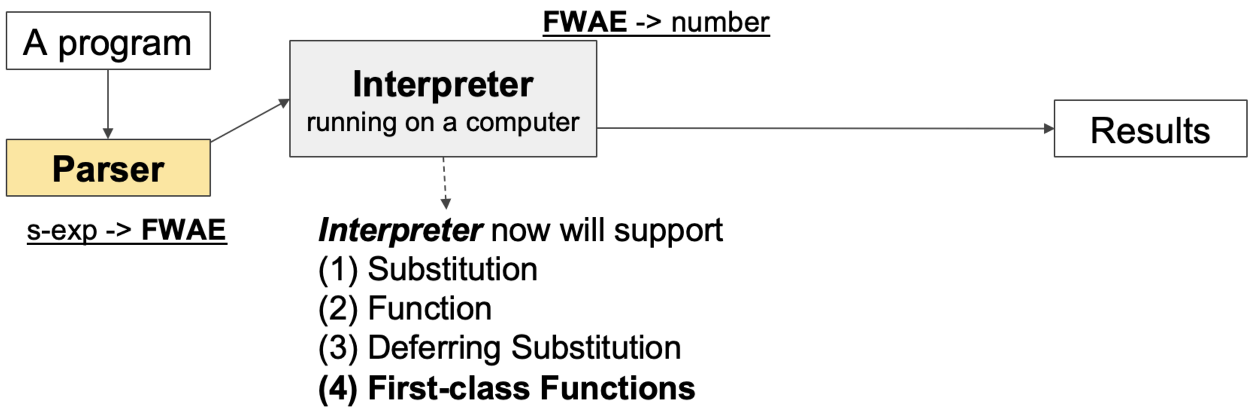
L10 & L11. First-class Function

유형	강의
복습 여부	✓
Status	Done



First-order Function

“Functions are not values in languages”

: Names must be given for use in the remainder of a program

Ex. F1WAE

First Class Function

- Functions can return other functions as values
- Pass function itself as parameter
- return function itself as a return value
- assign specific identifier for function definition
- define syntax for anonymous function that does not have any name but has parameter and body

```
{with {double {fun {x} {+ x x}} {+ {double 10} {double 5}}}
```

```
; Equivalent program by F1WAE
{deffun {f x} {+ x x}}
{+ {f 10} {f 5}}
```

FWAE

```
<FWAE> ::= <num>
         | {+ <FWAE> <FWAE>}
         | {- <FWAE> <FWAE>}
         | {with {<id> <FWAE>} <FWAE>}
         | <id>
         | {<FWAE> <FWAE>}
         | {fun {<id>} <FWAE>}
```

```
; FWAE that supports first-class functions
(define-type FWAE
  [num (n number?)]
  [add (lhs FWAE?) (rhs FWAE?)]
  [sub (lhs FWAE?) (rhs FWAE?)]
  [with (name symbol?) (named-expr FWAE?) (body FWAE?)]
  [id (name symbol?)]
  [fun (param symbol?) (body FWAE?)]
  [app (ftn FWAE?) (arg FWAE?)])
```

```
(fun 'x (add (id 'x) (id 'x)))
(app (fun 'x (add (id 'x) (id 'x))) (num 10))
```

```
(fun 'x (add (id 'x) (id 'x)))
(app (fun 'x (add (id 'x) (id 'x))) (num 10))
```

Parser

```
; [contract] parse : sexp -> FWAE
(define (parse sexp)
  (match sexp
    [(? number?) (num sexp)]
    [(list '+ l r) (add (parse l) (parse r))]
    [(list '- l r) (sub (parse l) (parse r))]
    [(list 'with (list i v) e) (with i (parse v) (parse e))]
    [(? symbol?) (id sexp)]
    [(list 'fun (list p) b) (fun p (parse b))]
    [(list f a) (app (parse f) (parse a))]
    [else (error 'parse "bad syntax:~a" sexp)]))
```

```
(parse '{fun {x} {+ x 1}})
(parse '{{fun {x} {+ x 1}} 10})
(parse '{with {x 3} {fun {x} {+ x y}}})
(parse '{with {x 3} {fun {y} {+ x y}}})
(parse '{with {z {fun {x} {+ x y}}} {with {y 10} z}})
```

```
(fun 'x (add (id 'x) (num 1)))
(app (fun 'x (add (id 'x) (num 1))) (num 10))
(with 'x (num 3) (fun 'x (add (id 'x) (id 'y))))
(with 'x (num 3) (fun 'y (add (id 'x) (id 'y))))
(with 'z (fun 'x (add (id 'x) (id 'y))) (with 'y (num 10) (id 'z)))
```

Interpreter

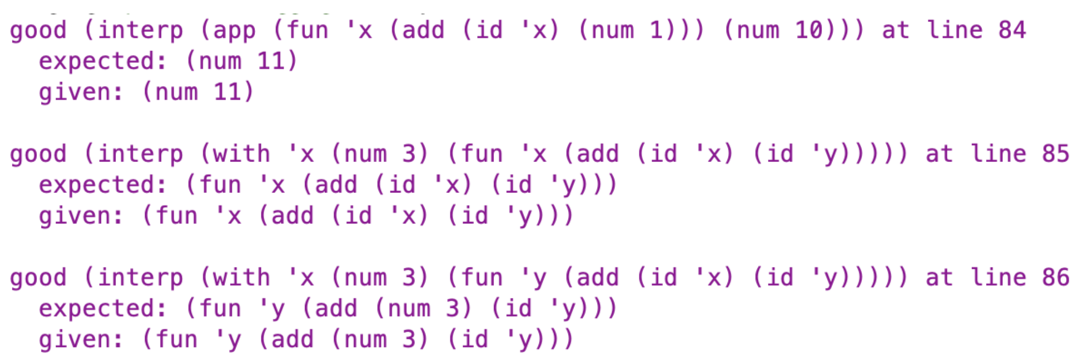
```
; [contract] interp : FWAE -> FWAE
(define (interp fwae)
  (type-case FWAE fwae
    [num (n) fwae]
    [add (l r) (num+ (interp l) (interp r))]
    [sub (l r) (num- (interp l) (interp r))]
    [with (i v e) (interp (subst e i (interp v)))]
    [id (s) (error 'interp "free identifier")]
    [fun (p b) fwae]
    [app (f a) (local
                 [(define ftn (interp f))]
                 (interp (subst (fun-body ftn)
                                (fun-param ftn)
                                (interp a)
                                )
                         )
                 )
         ]
    )
  )

; [contract] num-op : (number number -> number) -> (FWAE FWAE -> FWAE)
(define (num-op op)
  (lambda (x y)
    (num (op (num-n x) (num-n y)))
    )
  )

(define num+ (num-op +))
(define num- (num-op -))

; Substitution
; [contract] exp idtf val -> FWAE
(define (subst exp idtf val)
  (type-case FWAE exp
    [num (n) exp]
    [add (l r) (add (subst l idtf val) (subst r idtf val))]
    [sub (l r) (sub (subst l idtf val) (subst r idtf val))]
    [with (i v e) (with i (subst v idtf val)
                        (if (symbol=? i idtf) e (subst e idtf val)))]
    [id (s) (if (symbol=? s idtf) val exp)]
    [app (f a) (app (subst f idtf val) (subst a idtf val))]
    [fun (id body) (if (equal? idtf id) exp (fun id (subst body idtf val)))]
    )
  )
```

```
; Test cases provided in PPT
(test (interp (app (fun 'x (add (id 'x) (num 1))) (num 10))) (num 11))
(test (interp (with 'x (num 3) (fun 'x (add (id 'x) (id 'y))))) (fun 'x (add (id 'x) (id 'y))))
(test (interp (with 'x (num 3) (fun 'y (add (id 'x) (id 'y))))) (fun 'y (add (num 3) (id 'y))))
```



```
good (interp (app (fun 'x (add (id 'x) (num 1))) (num 10))) at line 84
  expected: (num 11)
  given: (num 11)

good (interp (with 'x (num 3) (fun 'x (add (id 'x) (id 'y))))) at line 85
  expected: (fun 'x (add (id 'x) (id 'y)))
  given: (fun 'x (add (id 'x) (id 'y)))

good (interp (with 'x (num 3) (fun 'y (add (id 'x) (id 'y))))) at line 86
  expected: (fun 'y (add (num 3) (id 'y)))
  given: (fun 'y (add (num 3) (id 'y)))
```

```
(test (interp (with 'x (num 5) (add (id 'x) (id 'x)))) (num 10))
(test (interp (with 'x (num 5) (add (num 1) (with 'y (id 'x) (id 'y))))) (num 6))
(test (interp (parse '{fun {a} {+ a a}})) (fun 'a (add (id 'a) (id 'a))))
(test (interp (parse '{with {fn {fun {a} {+ a a}}} {with {x 1} {fn {with {y 10} {+ y x}}}}})) (num 22))
```

```
good (interp (with 'x (num 5) (add (id 'x) (id 'x)))) at line 89
  expected: (num 10)
  given: (num 10)

good (interp (with 'x (num 5) (add (num 1) (with 'y (id 'x) (id 'y))))) at line 90
  expected: (num 6)
  given: (num 6)

good (interp (parse '(fun (a) (+ a a)))) at line 91
  expected: (fun 'a (add (id 'a) (id 'a)))
  given: (fun 'a (add (id 'a) (id 'a)))

good (interp (parse '(with (fn (fun (a) (+ a a))) (with (x 1) (fn (with (y 10) (+ y x))))))) at line 92
  expected: (num 22)
  given: (num 22)
```

Issues with the current Interpreter

Dynamic Scope

```
(interp (parse '{with {y 3} {with {z {fun {x} {+ x y}}} {with {y 10} z}}})) ; works fine
(interp (parse '{with {z {fun {x} {+ x y}}} {with {y 10} z}})) ; must produce error because 'y' in {fun {x} {+ x y}} is a free identifier
(interp (parse '{with {y 3} {with {z {fun {x} {+ x y}}} {with {y 10} {z 5}}}})) ; works fine
(interp (parse '{with {z {fun {x} {+ x y}}} {with {y 10} {z 5}}})) ; ; must produce error because 'y' in {fun {x} {+ x y}} is a free identi

; We are adopting static scope.
; If we use deferred substitution, we can solve the dynamic scope issue.
```

```
(fun 'x (add (id 'x) (num 3)))
(fun 'x (add (id 'x) (num 10)))
(num 8)
(num 15)
```