♣

L7. Substitution 2

유형	강의
복습 여부	☑
Status	Done

WAE and its interpretation

(with 'x (num 5) (add (id 'x) (id 'x)) [Substitution]
(with 'x (num 5) (add (num 5) (num 5)) [Descend]
(add (num 5) (num 5)) [add operator]
10 [Result]

We need an interpreter function as well as a substitution function.

Implement substitution for WAE Interpreter

```
; [contract] subst : WAE symbol number -> WAE
; (here, symbol is an identifier and number is the value for the identifier)
; [purpose] substitute second argument with third argument in first argument
; according to the rules of substitution, the resulting expression contains no free instances of the second argument
(define (subst wae idtf val)
  (type-case WAE wae
    [num (n) wae]
    [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) (num val) wae)]))
```

```
; {with {x 10} 5} => 10 for x in 5 => 5 (no substitution)
(test (subst (num 5) 'x 10) (num 5))
; {with {x 10} {+ 1 x}} => 10 for x in {+ 1 x} => {+ 1 10}
(test (subst (add (num 1) (id 'x)) 'x 10) (add (num 1) (num 10)))
; {with {x 10} x} => 10 for x in x => 10
(test (subst (id 'x) 'x 10) (num 10))
; {with {x 10} y} => 10 for x in y => y (no subsitution)
(test (subst (id 'y) 'x 10) (id 'y))
; {with {y 10} {-x 1}} => 10 for y in {-x 1} => {- x 1} (no subsitution)
(test (subst (sub (id 'x) (num 1)) 'y 10) (sub (id 'x) (num 1)))
```

```
; {with {x 10} {with {y 17} x}} => 10 for x in {with {y 17} x} => {with {y 17} 10}
(test (subst (with 'y (num 17) (id 'x)) 'x 10) (with 'y (num 17) (num 10)))
; {with {x 10} {with {y x} y}} => 10 for x in {with {y x} y} => {with {y 10} y}
(test (subst (with 'y (id 'x) (id 'y)) 'x 10) (with 'y (num 10) (id 'y)))
; {with {x 10} {with {x y} x}} => 10 for x in {with {x y} x} => {with {x y} x}
(test (subst (with 'x (id 'y) (id 'x)) 'x 10) (with 'x (id 'y) (id 'x)))
```

```
good (subst (num 5) 'x 10) at line 39
  expected: (num 5)
  given: (num 5)

good (subst (add (num 1) (id 'x)) 'x 10) at line 41
  expected: (add (num 1) (num 10))
  given: (add (num 1) (num 10))

good (subst (id 'x) 'x 10) at line 43
  expected: (num 10)
  given: (num 10)

good (subst (id 'y) 'x 10) at line 45
  expected: (id 'y)
  given: (id 'y)

good (subst (sub (id 'x) (num 1)) 'y 10) at line 47
  expected: (sub (id 'x) (num 1))
  given: (sub (id 'x) (num 1))

good (subst (with 'y (num 17) (id 'x)) 'x 10) at line 50
  expected: (with 'y (num 17) (num 10))
  given: (with 'y (num 17) (num 10))

good (subst (with 'y (id 'x) (id 'y)) 'x 10) at line 52
  expected: (with 'y (num 10) (id 'y))
  given: (with 'y (num 10) (id 'y))

good (subst (with 'x (id 'y) (id 'x)) 'x 10) at line 54
  expected: (with 'x (id 'y) (id 'x))
  given: (with 'x (id 'y) (id 'x))
```

Implement WAE Interpreter

```
; [contract] interp : WAE -> number
(define (interp wae)
  (type-case WAE wae
    [num (n) n]
    [add (l r) (+ (interp l) (interp r))]
    [sub (l r) (- (interp l) (interp r))]
    [with (i v e) (interp (subst e i (interp v)))]
    [id (s) (error 'interp "free identifier")]))
```

```
; {with {x 5} {+ x x}}
(test (interp (with 'x (num 5) (add (id 'x) (id 'x)))) 10)
; {with {x 5} {+ 1 {with {y x} y}}}
(test (interp (with 'x (num 5) (add (num 1) (with 'y (id 'x) (id 'y))))) 6)
; {with {x 10} y}
(test/exn (interp (with 'x (num 10) (id 'y))) "interp: free identifier")
```

```
good (interp (with 'x (num 5) (add (id 'x) (id 'x)))) at line 51
  expected: 10
  given: 10

good (interp (with 'x (num 5) (add (num 1) (with 'y (id 'x) (id 'y))))) at line 52
  expected: 6
  given: 6

good (interp (with 'x (num 10) (id 'y))) at line 53
  expected: "interp: free identifier"
  given: "interp: free identifier"
```