arithmetic operators

What is the result of the following program?

```
(deffun (f o) (o 1 1))
(f +)
```

- ○ Error
- ○ 2
- ○ Syntax error
- ○ Other

Please specify

0 as condition

What is the result of the following program?

(if 0 #t #f)

- ○ #f
- ○ #t
- ○ Other

Please specify

redeclare var using defvar

What is the result of the following program?

```
(defvar x 0)
(defvar y x)
(defvar x 2)
x
y
```

- ○ Error
- ○ 0; 0
- ○ 2; 0
- ○ Nothing is printed
- ○ Other

Please specify

expose local defvar

What is the result of the following program?

```
(defvar x 42)

(deffun (create)
  (defvar y 42)
  y)

(create)
(equal? x y)
```

- ○ Error
- ○ 42; #t
- ○ Other

Please specify

pair?

What is the result of the following program?

```
(pair? (pair 1 2))
(pair? (ivec 1 2))
(pair? '#(1 2))
(pair? '(1 2))
```

- ○ #t; #f; #t; #f
- ○ #t; #t; #t; #t
- ○ #t; #t; #t; #f
- ○ Other

Please specify

let* and let

What is the result of the following program?

```
(let* ([v 1]
       [w (+ v 2)]
       [y (* w w)])
  (let ([v 3]
        [y (* v w)])
    y))
```

- ○ 27
- ○ 9
- ○ Other

Please specify

defvar and let

What is the result of the following program?

```
(defvar x 3)
(defvar y (let ([y 6] [x 5]) x))

(* x y)
```

- ○ 9
- ○ 25
- ○ Other

Please specify

fun-id equals to arg-id

What is the result of the following program?

```
(deffun (f f) f)
(f 5)
```

- Error
- 5
- Other

Please specify

scoping rule of let

What is the result of the following program?

```
(let ([x 4]
      [y (+ x 10)])
  y)
```

- Error
- 14
- Other

Please specify

the right component of ivec

What is the result of the following program?

```
(right (ivec 1 2 3))
```

- ○ Error
- ○ 2
- ○ Other

Please specify

identifiers

What is the result of the following program?

```
(defvar x 5)


(deffun (reassign var_name new_val)
  (defvar var_name new_val)
  (pair var_name x))


(reassign x 6)


x
```

- '#(6 5); 5
- Nothing is printed
- '#(6 6); 5
- Error
- '#(6 6); 6
- Other

Please specify

defvar, deffun, and let

What is the result of the following program?

```
(defvar a 1)

(deffun (what-is-a) a)

(let ([a 2])
  (ivec
    (what-is-a)
    a))
```

- ‘#(1 2)
- ‘#(2 2)
- Other

Please specify

syntax pitfall

What is the result of the following program?

```
(deffun (f a b) a + b)

(f 5 10)
```

- 15
- 5
- Error
- Other

Please specify