Transforming a Point

Convert Avenues corners to SLC coordinates

~~`; ave->slc : string num -> num num`~~

Must return a single value

Correct contract:

```
; ave->slc : string num -> posn
```

A `posn` is a ***compound value***

Positions

- A **posn** is

  (make-posn X Y)

  where **X** is a **num** and **Y** is a **num**

Examples:

```
(make-posn 1 2)

(make-posn 17 0)
```

A **posn** is a value, just like a number, symbol, or image

posn-x and posn-y

The **posn-x** and **posn-y** operators extract numbers from a **posn**:

```
(posn-x (make-posn 1 2)) → 1

(posn-y (make-posn 1 2)) → 2
```

- General evaluation rules for any values **X** and **Y**:

```
(posn-x (make-posn X Y)) → X

(posn-y (make-posn X Y)) → Y
```

Positions and Values

Is `(make-posn 100 200)` a value?

Yes.

A `posn` is

`(make-posn X Y)`

where `X` is a `num` and `Y` is a `num`

Positions and Values

Is `(make-posn (+ 1 2) 200)` a value?

No. `(+ 1 2)` is not a `num`, yet.

- Two more evaluation rules:

`(make-posn X Y)` → `(make-posn Z Y)`
when `X` → `Z`

`(make-posn X Y)` → `(make-posn X Z)`
when `Y` → `Z`

Example:

```
(make-posn (+ 1 2) 200) →
(make-posn 3 200)
```

More Examples

Try these in DrRacket's stepper:

```
(make-posn (+ 1 2) (+ 3 4))

(posn-x (make-posn (+ 1 2) (+ 3 4)))

; pixels-from-corner : posn -> num
(define (pixels-from-corner p)
  (+ (posn-x p) (posn-y p)))
(pixels-from-corner (make-posn 1 2))

; flip : posn -> posn
(define (flip p)
  (make-posn (posn-y p) (posn-x p)))
(flip (make-posn 1 2))
```