Left and Right

Directions: Use the Design Recipe to write a function `safe-left?`, which takes in an x-coordinate and checks to see if it is greater than -50.

Contract and Purpose Statement

Every contract has three parts…

; `safe-left?` : `Number` -> `Boolean`

function name | *domain* | *range*

; Consumes an x-coordinate, checks to see if it is greater than -50, and produces a Boolean

what does the function do?

Examples

Write some examples, then circle and label what changes…

(EXAMPLE (`safe-left?` `100`) (`> 100 -50`))

function name | *input(s)* | *what the function produces*

(EXAMPLE (`safe-left?` `-180`) (`> -180 -50`))

function name | *input(s)* | *what the function produces*

Definition

Write the definition, giving variable names to all your input values…

(define (`safe-left?` `x`)

function name | *variable(s)*

(`> x -50`))

what the function does with those variable(s)

Directions: Use the Design Recipe to write a function `safe-right?`, which takes in an x-coordinate and checks to see if it is less than 690.

Contract and Purpose Statement

Every contract has three parts…

; `safe-right?` : `Number` -> `Boolean`

function name | *domain* | *range*

; Consumes an x-coordinate, checks to see if it is less than 690, and produces a Boolean

what does the function do?

Examples

Write some examples, then circle and label what changes…

(EXAMPLE (`safe-right?` `250`) (`< 250 690`))

function name | *input(s)* | *what the function produces*

(EXAMPLE (`safe-right?` `900`) (`< 900 690`))

function name | *input(s)* | *what the function produces*

Definition

Write the definition, giving variable names to all your input values…

(define (`safe-right?` `x`)

function name | *variable(s)*

(`< x 690`))

what the function does with those variable(s)