Left and Right

Directions: Use the Design Recipe to write a function `is-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…

```
# is-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…

examples:

```
is-safe-left ( 100 ) is 100 > -50
```
function name *input(s)* *what the function produces*

```
is-safe-left ( -180 ) is -180 > -50
```
function name *input(s)* *what the function produces*

end

Definition

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

```
fun is-safe-left( x ):
```
function name *variable(s)*

```
  x > -50
```
what the function does with those variable(s)

end

Directions: Use the Design Recipe to write a function `is-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…

```
# is-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…

examples:

```
is-safe-right ( 250 ) is 250 < 690
```
function name *input(s)* *what the function produces*

```
is-safe-right ( 900 ) is 900 < 690
```
function name *input(s)* *what the function produces*

end

Definition

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

```
fun is-safe-right( x ):
```
function name *variable(s)*

```
  x < 690
```
what the function does with those variable(s)

end