Name: _____

Bootstrap:2

www.bootstrapworld.org

Class: _____

Workbook v0.9

Brought to you by the Bootstrap team:

- Emma Youndtsmith
- Emmanuel Schanzer
- Kathi Fisler
- Shriram Krishnamurthi

Visual Design: Colleen Murphy

Lesson 1

	Racket Code	Pyret Code
Numbers	(define AGE 14) (define A-NUMBER 0.6) (define SPEED -90)	AGE = 14 A-NUMBER = 0.6 SPEED = -90 Two of your own: **MY-NUMBER = 75.9** **THREE = 3**
Strings	(define CLASS “Bootstrap”) (define PHRASE "Coding is fun!") (define A-STRING "2500")	CLASS = “Bootstrap” PHRASE = “Coding is fun!” A-STRING = “2500” Two of your own: **MY-NAME = “Elizabeth”** **MY-NUMBER = 75.9**

	(define SHAPE
	  (triangle 40 "outline" "red"))
	
	(define OUTLINE
	  (star 80 "solid" "green"))
	
	(define SQUARE
	  (rectangle 50 50 "solid" "blue"))
		SHAPE =
		  triangle(40, "outline", "red")
		
		OUTLINE =
		  star(80, "solid", "green")
		
		SQUARE =
		  rectangle(50, 50, "solid", "blue")
		
		One of your own:
		
		MY-SHAPE = rhombus(90, 60, "solid", "red")

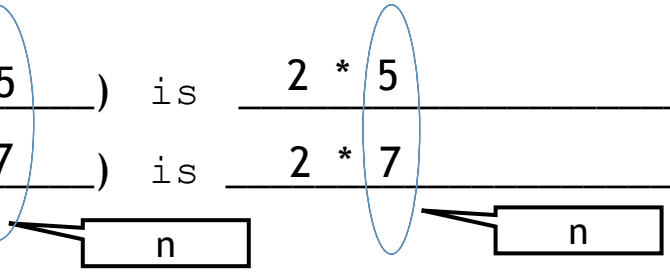
Booleans	(define BOOL true) (define BOOL2 false)	BOOL = true One of your own: **BOOL2 = false**

Functions	; double : Number -> Number
	; Given a number, multiply by
	; 2 to double it
	
	(EXAMPLE (double 5) (* 2 5)
	(EXAMPLE (double 7) (* 2 7))
	
	(define (double n) (* 2 n))
		# double : Number -> Number
		# Given a number, multiply by
		# 2 to double it
		
		examples:
		  double(5) is 2 * 5
		  double(7) is 2 * 7
		end
		
		fun double(n):
		  2 * n
		end

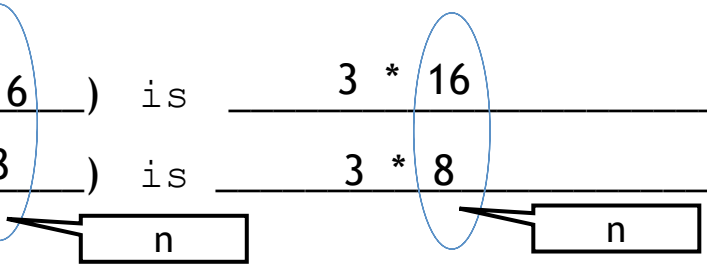
Fast Functions!

Fill out the contract for each function, then try to write two examples and the definition by yourself.

```
# double : Number -> Number
  name     domain    range

examples:
    double(5) is 2 * 5
    double(7) is 2 * 7
end

fun double(n):
    2 * n
end
```




```
# triple : Number -> Number
  name     domain    range

examples:
    triple(16) is 3 * 16
    triple(8) is 3 * 8
end

fun triple(n):
    3 * n
end
```

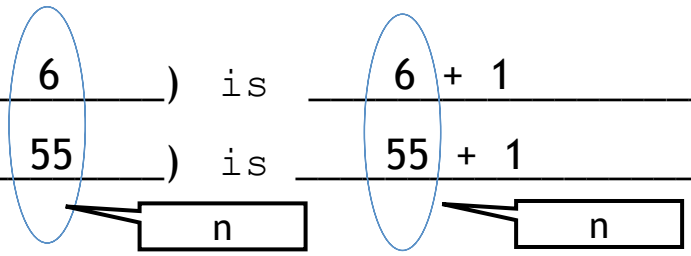


Fast Functions!

Fill out the contract for each function, then try to write two examples and the definition by yourself.

```
# plus1 : Number -> Number
  name   domain   range
```

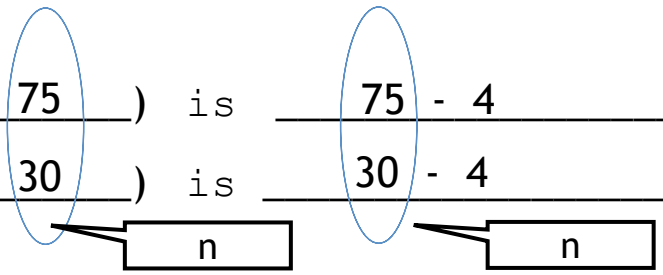
```
examples:
   plus1(6) is 6 + 1
   plus1(55) is 55 + 1
end
```




```
fun plus1(n):
    n + 1
end
```

```
# mystery : Number -> Number
  name     domain   range
```

```
examples:
   mystery(75) is 75 - 4
   mystery(30) is 30 - 4
end
```




```
fun mystery(n):
    n - 4
end
```

Fast Functions!

Fill out the contract for each function, then try to write two examples and the definition by yourself.

```
# red-spot : Number -> Image
  name     domain     range

examples:
   red-spot(20) is circle(20, "solid", "red")
   red-spot(99) is circle(99, "solid", "red")
end

fun red-spot(radius):
    circle(radius, "solid", "red")
end
```

radius

radius

```
# _____ : _____ -> _____
  name    domain    range

examples:
   _____(_____) is _____
   _____(_____) is _____
end

fun _____(_____):
    _____
end
```

Bug Hunting: Pyret Edition

#1	`SECONDS = (7)`
	
	`STRING = my string`
		SECONDS = 7
		
		STRING = “my string”

#2	`SHAPE1 = circle(50 “solid” “blue”)`
	
	`SHAPE2 = triangle(75, outline, yellow)`
		SHAPE1 = circle(50, “solid”, “blue”)
		
		SHAPE2 = triangle(75, “outline”, “yellow”)

#3	`# triple : Number -> Number`
	`# Multiply a given number by`
	`# 3 to triple it`
	
	`examples:`
	`  triple(5) = 3 * 5`
	`  triple(7) = 3 * 7`
	`end`
		# triple : Number -> Number
		# Multiply a given number by 3 to triple it
		
		examples:
		** triple(5) is 3 * 5**
		** triple(7) is 3 * 7**
		end

#4	`fun triple(n):` `  3 * n`	**fun triple(n) :** ** 3 * n** **end**

#5	`# ys : Number -> Number`
	`# Given a number, create a solid`
	`# yellow star of the given size`
	
	`examples:`
	`  ys(99) is star(99, “solid”, “yellow”)`
	`  ys(33) is star(99, “solid”, “yellow”)`
	
	`ys(size):`
	`  star(size “solid” “yellow”)`
	`end`
		# ys : Number -> Number
		# Given a number, create a solid yellow star of the given size
		
		examples:
		** ys(99) is star(99, “solid”, “yellow”)**
		** ys(99) is star(99, “solid”, “yellow”)**
		end
		
		ys(size) :
		** star(size, “solid”, “yellow”)**
		end

Lesson 2

Word Problem: double-radius

Write a function *double-radius*, which takes in a radius and a color. It produces an outlined circle of whatever color was passed in, whose radius is twice as big as the input.

Contract+Purpose Statement

Every contract has three parts:

```
# double-radius : Number, String -> Image
    name          Domain           Range
# Consumes a number and a string, produces an outlined circle of the given color, whose
  radius is twice the given number
                What does the function do?
```

Give Examples

Write examples of your function in action

```
examples:
    double-radius (50, "pink")          is circle(50 * 2, "outline", "pink")
                   the user types...                ...which should become
    double-radius (918, "orange")       is circle(918 * 2, "outline", "orange")
                   the user types...                ...which should become
end
```

radius, color, radius

Function

Circle the changes in the examples, and name the variables.
Write the code, copying everything that isn't circled, and using names where you find variables!

```
fun double-radius (radius, color):

    circle(radius * 2, "outline", color)
end
```

Word Problem: double-width

Write a function *double-width*, which takes in a number (the length of a rectangle) and produces a solid green rectangle whose width is twice the given length.

Contract+Purpose Statement

Every contract has three parts:

```
# double-width : Number -> Image
  name           Domain    Range
```

```
# Consumes a length and produces a solid green rectangle whose width is twice the given length
  What does the function do?
```

Give Examples

Write examples of your function in action

```
examples:
    double-width (45)        is
      the user types...
    rectangle(45, 45 * 2, "solid", "green")
      ...which should become

    double-width (8)         is
      the user types...
    rectangle(8, 8 * 2, "solid", "green")
      ...which should become
end
```

length

length

Function

Circle the changes in the examples, and name the variables.
Write the code, copying everything that isn't circled, and using names where you find variables!

```
fun double-width (length):

    rectangle(length, length * 2, "solid", "green")
end
```

Word Problem: next-position

Write a function *next-position*, which takes in two numbers (an x and y-coordinate) and returns a Coord, increasing the x-coordinate by 5 and decreasing the y-coordinate by 5.

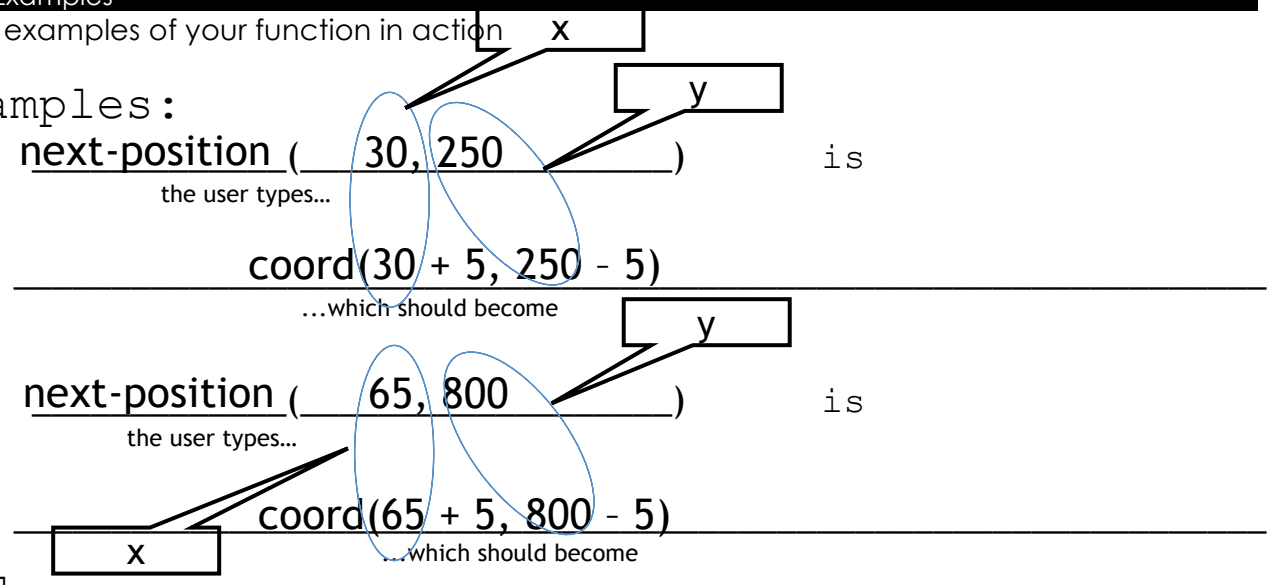
Contract+Purpose Statement

Every contract has three parts:

```
# next-position : Number, Number -> Coord
  name          Domain             Range

#Given 2 numbers, make a Coord by adding 5 to x and subtracting 5 from y
                What does the function do?
```

Give Examples

Write examples of your function in action



Function

Circle the changes in the examples, and name the variables.
Write the code, copying everything that isn't circled, and using names where you find variables!

```
fun next-position ( x , y ):

    coord(x + 5, y - 5)
end
```

Data Structure

```
# a Cake is a flavor, color, message, layers, & is-iceCream
data Cake:
    | cake(flavor :: String,
           color  :: String,
           message :: String,
           layers :: Number,
           is-iceCream :: Boolean)
end
```

To make examples of this structure, I would write:

cake1 = cake("Vanilla", "white", "Happy wedding!", 4, false)

cake2 = cake("Red Velvet", "darkred", "I love cakes!", 2, true)

To access the fields of **cake2**, I would write:

cake2.flavor

cake2.color

cake2.message

cake2.layers

cake2.is-iceCream

Lesson 3

Data Structure

```
# a Party is a location, theme, and number of guests
data Party:
    | party(location :: String,
           theme :: String,
           guests :: Number)
end
```

To make examples of this structure, I would write:

party1 = party("Downtown", "80s", 34)

party2 = party("bowling ally", "bowling", 20)

To access the fields of **party2**, I would write:

party2.location

party2.theme

party2.guests

Word Problem: change-flavor

Write a function called *change-flavor,* which takes in a Cake and a flavor, and returns a new Cake that is almost the same as the original, but is now the given flavor.

Contract+Purpose Statement

```
# change-flavor : Cake, String -> Cake
# Given a Cake and a flavor, return a new Cake that is the same as the original, but with the
  given flavor
```

Give Examples

```
examples:
  change-flavor(cake1, "strawberry") is
                cake("strawberry",
                     cake1.color,
                     cake1.message,
                     cake1.layers,
                     cake1.is-iceCream)
  change-flavor(cake2, "vanilla") is
                cake("vanilla",
                     cake2.color,
                     cake2.message,
                     cake2.layers,
                     cake2.is-iceCream)
end
```

a-cake

a-cake

a-cake

a-cake

Function

```
fun change-flavor(a-cake, new-flavor) :
    cake(new-flavor,
         a-cake.color,
         a-cake.message,
         a-cake.layers,
         a-cake.is-iceCream)
end
```

Word Problem: will-melt

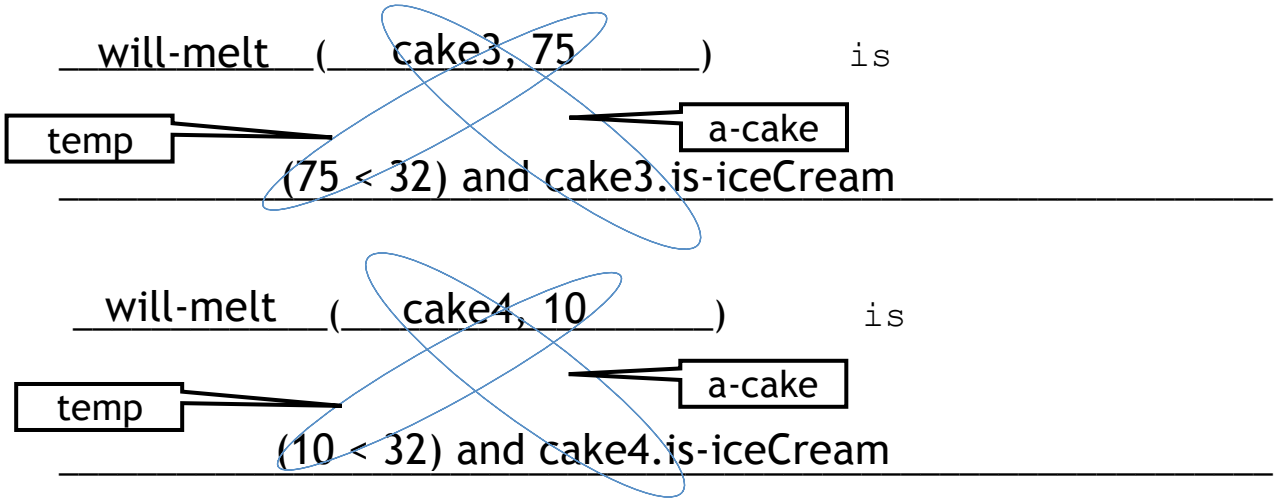
Write a function called *will-melt,* which takes in a Cake and a temperature, and returns true if the temperature is greater than 32 degrees, AND the Cake is an ice cream cake.

Contract+Purpose Statement

```
# will-melt : Cake, Number -> Boolean
# Given a Cake and a temperature, return true if the temp is greater than 32 degrees,
  AND the Cake is an ice cream cake
```

Give Examples

```
examples:
  will-melt(cake3, 75) is
    (75 < 32) and cake3.is-iceCream

  will-melt(cake4, 10) is
    (10 < 32) and cake4.is-iceCream
end
```



Function

```
fun will-melt(a-cake, temp):

  (temp < 32) and a-cake.is-iceCream
end
```

Lesson 4

Lesson 5

Word Problem: keypress (Ninja World)

State the Problem

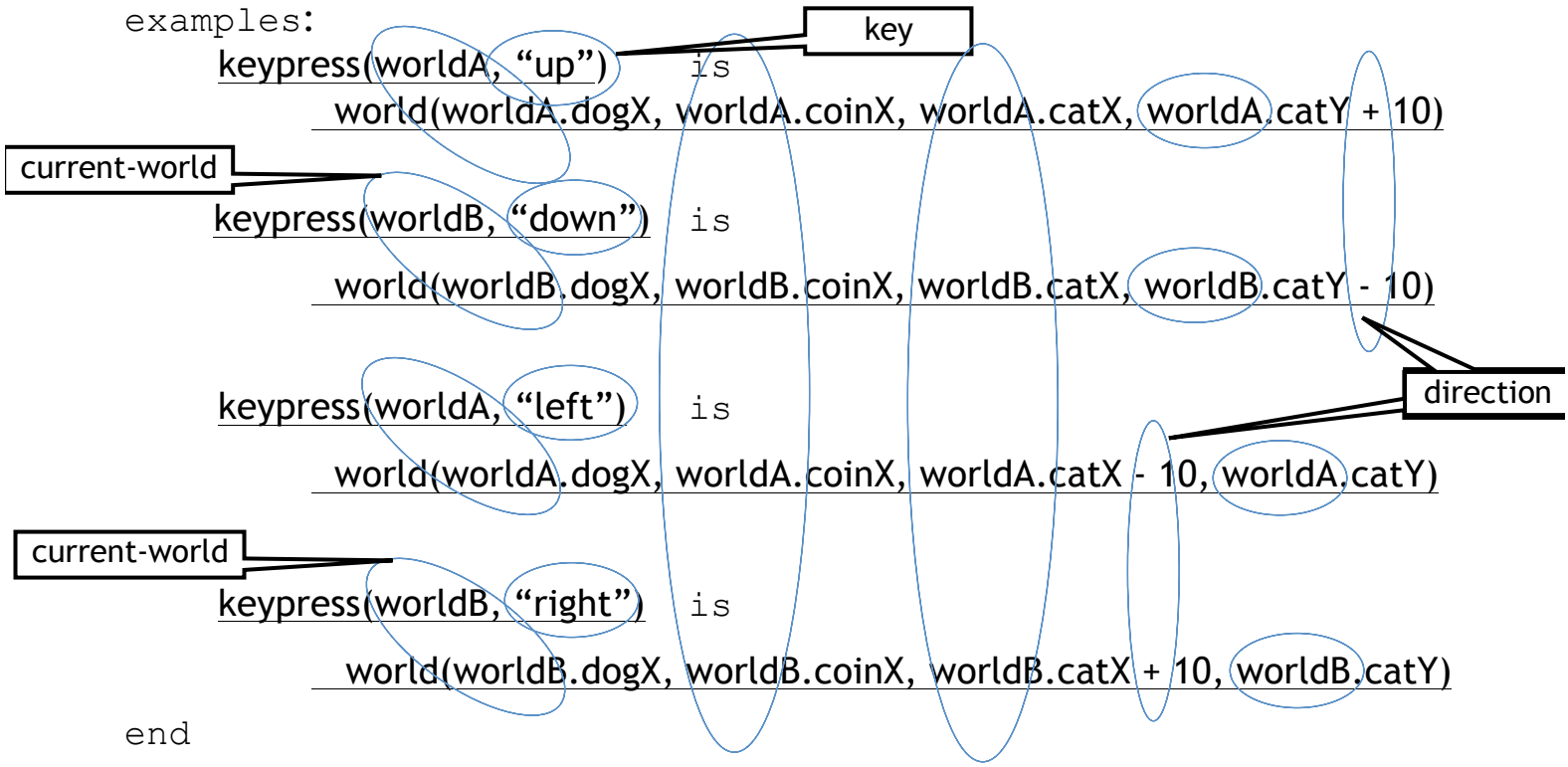
For each keypress in Ninja World, show how (keypress <world > <key>) should change the world.

Contract+Purpose Statement

```
# keypress : World, String -> World

# Given a world and a key, produce a new world with NinjaCat's position
# moved by 10 pixels, depending on which arrow key was pressed
```

Give Examples

```
examples:
    keypress(worldA, "up")    is
      world(worldA.dogX, worldA.coinX, worldA.catX, worldA.catY + 10)

    keypress(worldB, "down")  is
      world(worldB.dogX, worldB.coinX, worldB.catX, worldB.catY - 10)

    keypress(worldA, "left")  is
      world(worldA.dogX, worldA.coinX, worldA.catX - 10, worldA.catY)

    keypress(worldB, "right") is
      world(worldB.dogX, worldB.coinX, worldB.catX + 10, worldB.catY)
end
```



```
fun keypress(current-world, key) :

    ask:
     | string-equal(key, “up”) then:

          world(current-world.dogX, current-world.coinX,
                current-world.catX, current-world.catY + 10)

     | string-equal(key, “down”) then:

          world(current-world.dogX, current-world.coinX,
                current-world.catX, current-world.catY - 10)

     | string-equal(key, “left”) then:

          world(current-world.dogX, current-world.coinX,
                current-world.catX - 10, current-world.catY)

     | string-equal(key, “right”) then:

          world(current-world.dogX, current-world.coinX,
                current-world.catX + 10, current-world.catY)

     | otherwise: current-world

    end
end
```

Word Problem: next-world (Ninja World)

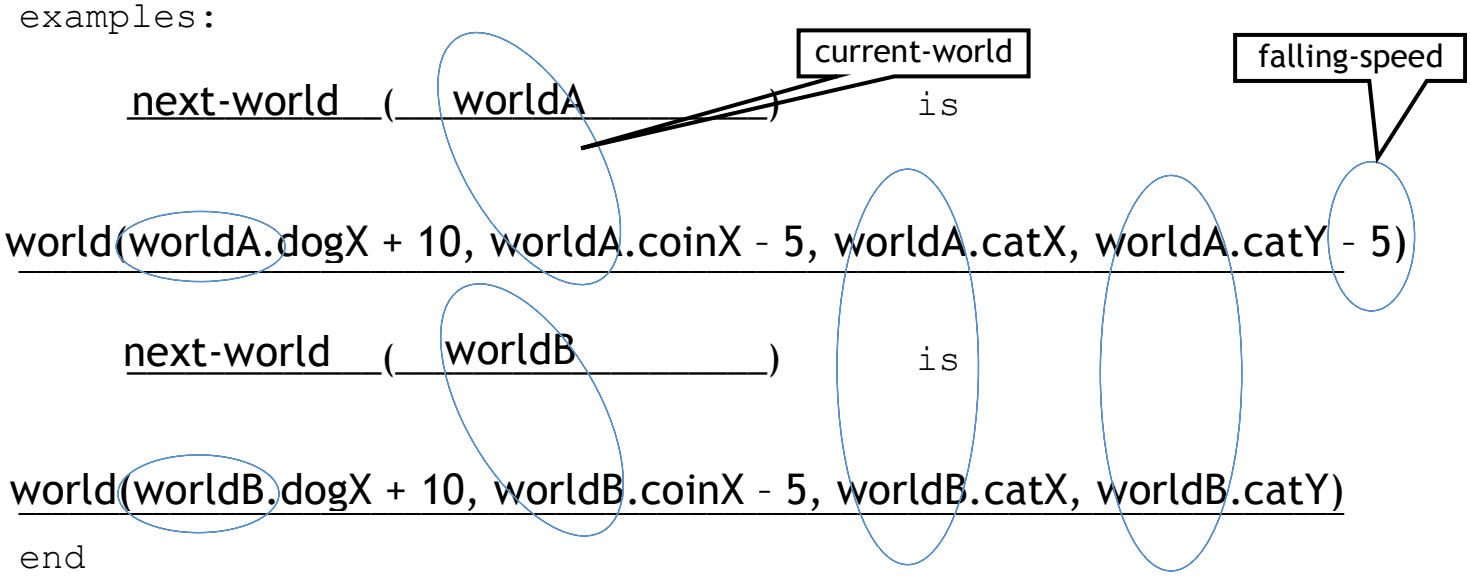
Given a world, return the next world by adding 10 to dogX, subtracting 5 from coinX, and subtracting 5 from catY *only* when the cat's y-coordinate is greater than 75.

Contract+Purpose Statement

```
# next-world : World -> World
# Given a World, check whether CatY is greater than 75. If so, create a world by
  adding 10 to dogX and subtracting 5 from coinX and catY. Otherwise, create a world
  whose catY is the same as the current world, with dogX and coinX changing as above
```

Give Examples

```
examples:

  next-world (worldA) is
  world(worldA.dogX + 10, worldA.coinX - 5, worldA.catX, worldA.catY - 5)

  next-world (worldB) is
  world(worldB.dogX + 10, worldB.coinX - 5, worldB.catX, worldB.catY)
end
```



Function

```
fun next-world(current-world) :
  ask:
    | current-world.catY > 75 then:
world(current-world.dogX + 10, current-world.coinX - 5, current-world.catX, current-world.catY - 5)
    | otherwise:
world(current-world.dogX + 10, current-world.coinX - 5, current-world.catX, current-world.catY)
  end
end
```

Lesson 6

Word Problem: red-shape

Write a function *red-shape*, which takes in the name of a shape (such as "circle", "triangle", "star", or "rectangle"), and draws that solid, red shape. Use 50 as the radius of the circle and star, and side-length of the triangle. Make the rectangle 99 pixels long by 9 wide.

```
# red-shape : String -> Image
# Consumes the name of a shape, and produces a solid, red image of that
# shape. Use 50 for size of the circle, star, and triangle, and make the
  rectangle 99 x 9
```

Give Examples

```
examples:
  red-shape ( "circle" ) is circle( 50, "solid", "red")
  red-shape ( "triangle" ) is triangle( 50, "solid", "red")
  red-shape ( "star" ) is star( 50, "solid", "red")
  red-shape ( "rectangle" ) is rectangle( 99, 9, "solid", "red")
end
```

shape

shape-name

size

Function

```
fun red-shape ( shape ) :
  ask:
    | string-equal(shape, "circle") then:
      circle( 50, "solid", "red")
    | string-equal(shape, "triangle then:
      triangle( 50, "solid", "red")
    | string-equal(shape, "star") then:
      star( 50, "solid", "red")
    | string-equal(shape, "rectangle") then:
      rectangle( 99, 9, "solid", "red")
  end
end
```

Word Problem: strong-password

Websites have strict password requirements. Write a function *strong-password*, which takes in a username and password, and checks to make sure they aren't the same, and then checks the string-length of the password to make sure it is greater than 8 characters. The function should return a message to the user letting them know if their password is strong enough.

strong-password : String, String -> String
Given a username and password, check whether they are the same, then check whether the string-length of the password is greater than 8

Give Examples

```
examples:
```

username

password

strong-password ("Coolguy90", "Coolguy90") is "Your username cannot be the same as your password!"

message

strong-password ("greatname", "abc") is "Your password is too short! Must be at least 8 characters."

strong-password ("Katie", "BootstrapPro78") is "Your password is strong enough! Account created."

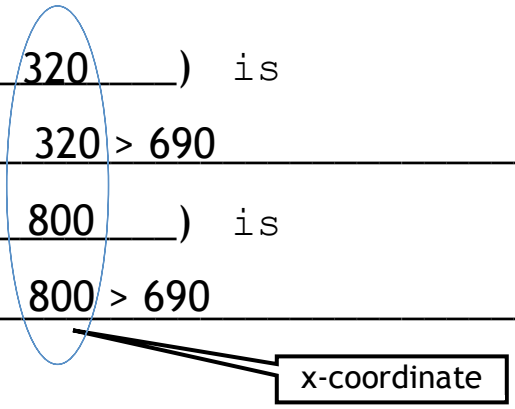
```
end
```

Function

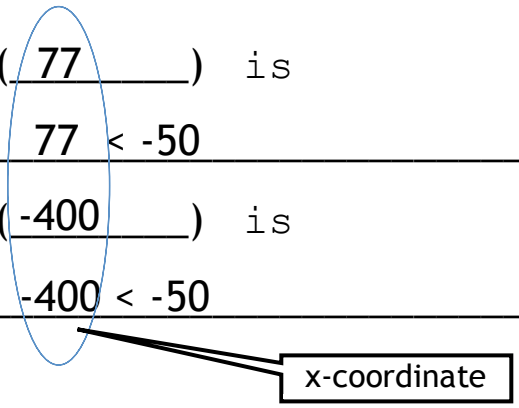
```
fun strong-password (username, password) :
    ask:
        | string-equal(username, password) then:
          "Your username cannot be the same as your password!"
        | string-length(password) < 8 then:
          "Your password is too short! Must be at least 8 characters."
        | otherwise: "Your password is strong enough! Account created."
    end
end
```

Building Your Helper Functions

```
# is-off-right : Number -> Boolean

examples:
   is-off-right (320) is
                320 > 690
   is-off-right (800) is
                800 > 690
end

fun is-off-right (x-coordinate):
    x-coordinate > 690
end
```



```
# is-off-left : Number -> Boolean

examples:
   is-off-left (77) is
               77 < -50
   is-off-left (-400) is
               -400 < -50
end

fun is-off-left (x-coordinate):
    x-coordinate < -50
end
```



```
# ___is-in-air_____:__Number_____ -> _Boolean_____
examples:
   ___is-in-air_____(_102____)  is
                ____102 > 75_____________________________
   ___is-in-air_____(_30_____)  is
                ____30 > 75______________________________
end
fun ___is-in-air___(_y-coordinate____):

   ____y-coordinate > 75_____________________________
end
```

y-coordinate

```
# _________________:_________________ -> _________________
examples:
   _____________________(_______)  is
                ____________________________________________
   _____________________(_______)  is
                ____________________________________________
end
fun ____________(_________________):

   ____________________________________________________________
end
```

Lesson 7

Word Problem: line-length

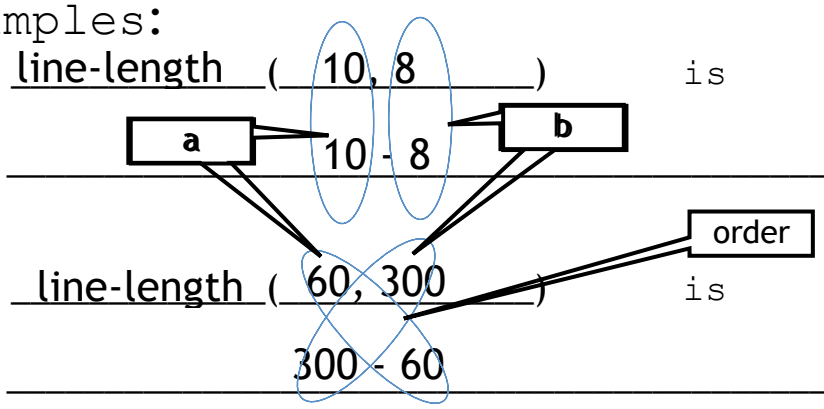
Write a function called *line-length*, which takes in two numbers and returns the difference between them. It should always subtract the smaller number from the bigger one.

Contract+Purpose Statement

```
# line-length : Number, Number -> Number
# Consumes 2 numbers and produces the difference by subtracting the smaller number from the larger
```

Give Examples

```
examples:
  line-length (10, 8) is
```




```
  line-length (60, 300) is
end
```

Function Header

```
fun line-length (a, b):
```

function name — variable names

```
  ask:
    | a > b then: a - b
    | otherwise: b - a
  end
end
```

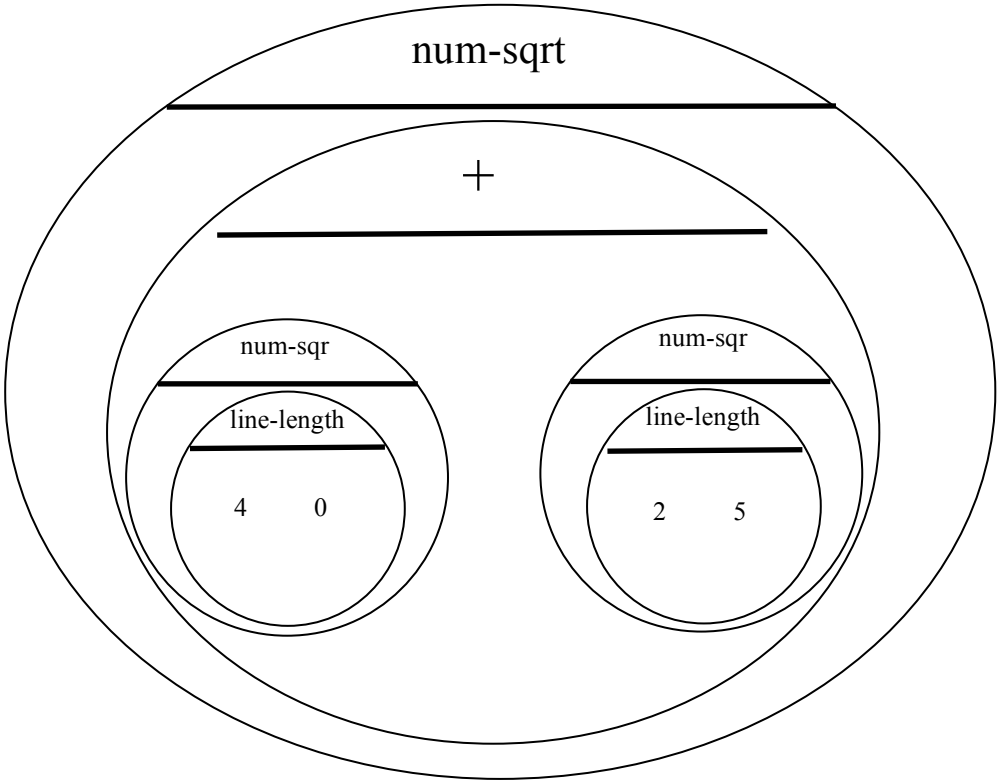
Distance:

The Player is at **(4, 2)** and the Target is at **(0, 5)**.
Distance takes in the player's x, player's y, character's x and character's y.

Use the formula below to fill in the EXAMPLE:

$$\sqrt{(line-length\ \ 4\ \ 0\ )^2\ +\ (line-length\ \ 2\ \ 5\ )^2}$$

Convert it into a Circle of Evaluation. (We've already gotten you started!)



Convert it into Pyret code:

num-sqrt(num-sqr(line-length(4, 0)) + num-sqr(line-length(2, 5)))

Word Problem: distance

Write a function `distance`*, which takes FOUR inputs:*

- *px: The x-coordinate of the player*
- *py: The y-coordinate of the player*
- *cx: The x-coordinate of* another game character
- *cy: The y-coordinate of another game character*

It should return the distance between the two, using the Distance formula:

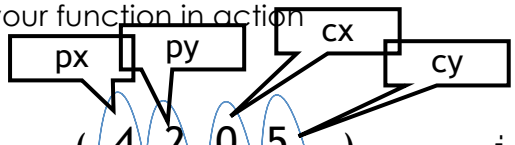
$$\text{Distance}^2 = (\text{line-length px cx})^2 + (\text{line-length py cy})^2$$

Contract+Purpose Statement

```
# distance : Number, Number, Number, Number -> Number
  Given the coordinates of 2 characters: px, py, cx, and cy, use the distance
# formula to calculate the distance between them
```

Give Examples

Write examples of your function in action




```
examples:
    distance(4, 2, 0, 5) is
  num-sqrt(num-sqr(line-length(4, 0)) + num-sqr(line-length(2, 5)))

    distance(80, 33, 6, 50) is
  num-sqrt(num-sqr(line-length(80, 6)) + num-sqr(line-length(33, 50)))
end
```

Function

```
fun distance(px, py, cx, cy):

  num-sqrt(num-sqr(line-length(px, cx)) + num-sqr(line-length(py, cy)))



end
```

Word Problem: is-collision

Write a function *is-collision*, which takes FOUR inputs:

- px: The x-coordinate of the player
- py: The y-coordinate of the player
- cx: The x-coordinate of another game character
- cy: The y-coordinate of another game character

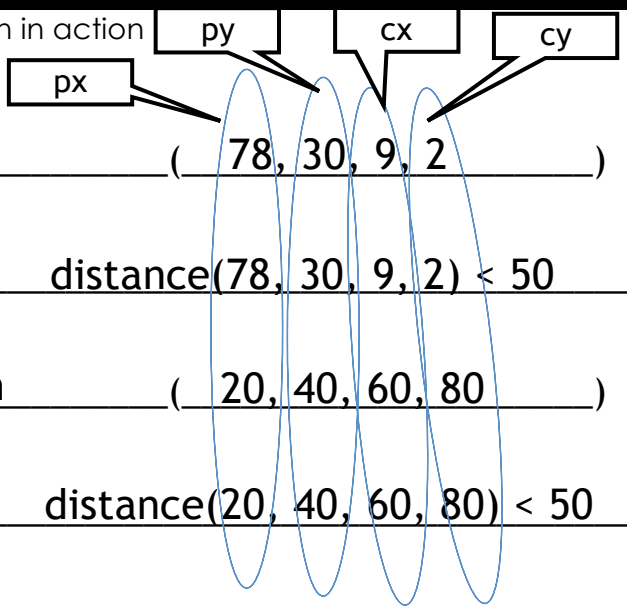
It should return true if the coordinates of the player are within **50 pixels** of the coordinates of the other character. Otherwise, false.

Contract+Purpose Statement

```
# is-collision : Number, Number, Number, Number -> Boolean
  Given the coordinates of 2 characters: px, py, cx, and cy, return true if the
# distance between them is less than 50 pixels
```

Give Examples

Write examples of your function in action




```
examples:
    is-collision(78, 30, 9, 2) is
    distance(78, 30, 9, 2) < 50
    is-collision(20, 40, 60, 80) is
    distance(20, 40, 60, 80) < 50
end
```

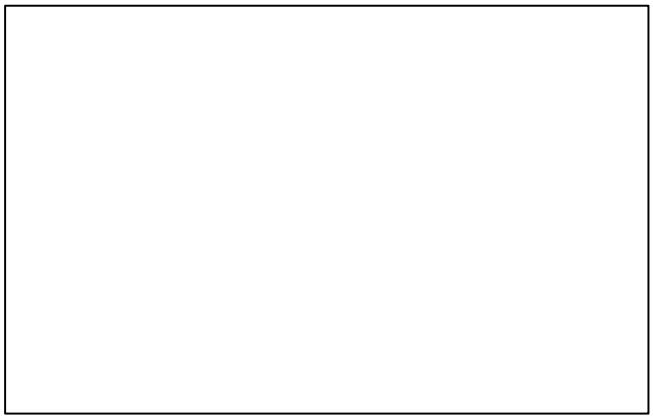
Function

```
fun is-collision(px, py, cx, cy):
    distance(px, py, cx, cy) < 50
end
```

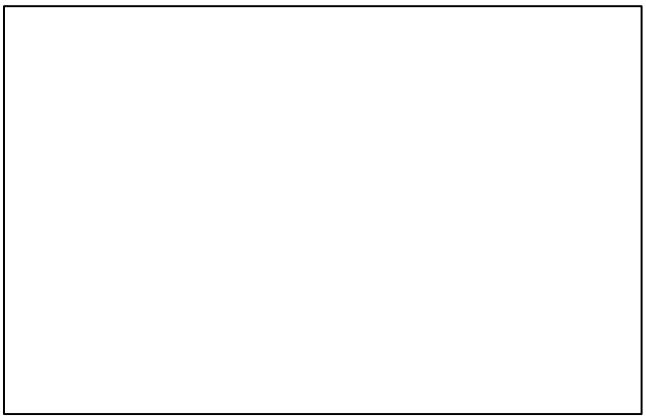
GAME DESIGN

"Start Simple, Get Complex"

Draw a rough sketch of your game when it begins, and another sketch just a moment later



A sketch at the START of the game…



A sketch for the very NEXT moment…

What images will you need for your game? Name them in the 1st column, and describe them in the 2nd

BACKGROUND	

List everything that has changed from one sketch to the other. What datatype will represent it?

Changed (position, score, color, costume…)	**Datatype** (Number, String, Image, Boolean…)

Data Structures

```
# a world is a _____________________________________
data World:
    | world(_____________________________________
            _____________________________________
            _____________________________________
            _____________________________________
            _____________________________________)
end
```

To make example worlds that represent my sketches from page 31, I would write…

worldA = _____

worldB = _____

To access the fields of **worldA**, I would write:

_____

_____

_____

_____

_____

Lesson 8

Word Problem: draw-world (My game)

Contract

_____ : _____ -> _____

Definition

```
fun _____(_____) :
  put-image(_____
            _____
            _____
            _____
            _____
            _____
            _____
            _____
            _____
            _____
            _____
            _____
            _____
            _____)
end
```

Word Problem: next-world (My game)

State the problem (What changes?):

Contract+Purpose Statement

_____ : _____ -> _____
_____

Give Examples

```
examples:
     _____(_____)       is

                _____
                _____
                _____
                _____
                _____
     _____(_____)       is

                _____
                _____
                _____
                _____
                _____
end
```

Function

```
fun _____(_____) :

                _____
                _____
                _____
                _____
                _____
end
```

Lesson 9

When *this* key is pressed…	..*this* field of the new world...	…changes by…

Word Problem: keypress (My game)

For each keypress in your game, show how `keypress(worldA, <key>)` should change your world.

: ->

#

Give Examples

`examples:`

keypress(worldA, _____) `is`

keypress(worldA, _____) `is`

keypress(worldA, _____) `is`

`end`

```
fun _____(_____)
    ask:
        | _____ then:
          _____
        | _____ then:
          _____
        | _____ then:
          _____
        | _____ then:
          _____
        | _____ then:
          _____
        | _____ then:
          _____
        |_____
    end
end
```

Building Your Helper Functions

is-off-right :_____ -> _____

```
examples:
  _____(_____) is
          _____
  _____(_____) is
          _____
end
fun _____(_____):
  _____
 end
```

---

is-off-left :_____ -> _____

```
examples:
  _____(_____) is
          _____
  _____(_____) is
          _____
end
fun _____(_____):
  _____
 end
```

```
# _____:_____ -> _____
examples:
  _____(_____) is
          _____
  _____(_____) is
          _____
end
fun _____(_____):

  _____
end
```

```
# _____:_____ -> _____
examples:
  _____(_____) is
          _____
  _____(_____) is
          _____
end
fun _____(_____):

  _____
end
```

Using Helpers inside `next-world`:

How does the World structure change when....?

TEST	RESULT
	world(_____ _____ _____ _____ _____)
	world(_____ _____ _____ _____ _____)
	world(_____ _____ _____ _____ _____)
	world(_____ _____ _____ _____ _____)

TEST	RESULT
	world(_____ _____ _____ _____ _____)
	world(_____ _____ _____ _____ _____)
	world(_____ _____ _____ _____ _____)
	world(_____ _____ _____ _____ _____)

Using Helpers inside `draw-world`:

What changes the *appearance* of your game?

TEST	RESULT
	put-image(_____ put-image(_____ put-image(_____ put-image(_____ put-image(_____
	put-image(_____ put-image(_____ put-image(_____ put-image(_____ put-image(_____
	put-image(_____ put-image(_____ put-image(_____ put-image(_____ put-image(_____
	put-image(_____ put-image(_____ put-image(_____ put-image(_____ put-image(_____

TEST	RESULT
	put-image(_____ put-image(_____ put-image(_____ put-image(_____ put-image(_____
	put-image(_____ put-image(_____ put-image(_____ put-image(_____ put-image(_____
	put-image(_____ put-image(_____ put-image(_____ put-image(_____ put-image(_____
	put-image(_____ put-image(_____ put-image(_____ put-image(_____ put-image(_____

Lesson 10

Supplemental

DESIGN RECIPE

Contract+Purpose Statement

Every contract has three parts:

_____ : _____ -> _____

name Domain Range

_____

What does the function do?

Give Examples

Write examples of your function in action

```
examples:
    _____(_____)        is
```

the user types…

_____

…which should become

_____(_____) is

the user types…

_____

…which should become

```
end
```

Function

Circle the changes in the examples, and name the variables.

```
fun _____(_____):


    _____
end
```

DESIGN RECIPE

Contract+Purpose Statement

Every contract has three parts:

_____ : _____ -> _____

name Domain Range

_____

What does the function do?

Give Examples

Write examples of your function in action

```
examples:
    _____(_____)        is
```

the user types…

_____

…which should become

_____(_____) is

the user types…

_____

…which should become

```
end
```

Function

Circle the changes in the examples, and name the variables.

```
fun _____(_____):


    _____
end
```

Contracts

Name	Domain	Range	example
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	

Contracts

Name	Domain	Range	example
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	
#	:	→	