Fast Functions!

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

Directions:

Contract and Purpose Statement
Every contract has three parts…

# __________ : __________ -> __________

# __________ : __________

Examples
Write some examples, then circle and label what changes…

examples:

double (5) is 2 * 5

double (7) is 2 * 7

end

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

fun __________ (__________):

__________

end

Directions:

Contract and Purpose Statement
Every contract has three parts…

# __________ : __________ -> __________

# __________ : __________

Examples
Write some examples, then circle and label what changes…

examples:

__________ (__________) is __________

__________ (__________) is __________

end

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

fun __________ (__________):

__________

end
Fast Functions!

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

Directions:

Contract and Purpose Statement

Every contract has three parts…

# : domain -> range

# what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces

end

Definition

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

fun (variable(s)):

what the function does with those variable(s)

end

Directions:

Contract and Purpose Statement

Every contract has three parts…

# : domain -> range

# what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces

end

Definition

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

fun (variable(s)):

what the function does with those variable(s)

end
Fast Functions!

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

Directions:

**Contract and Purpose Statement**

Every contract has three parts…

# function name :: domain -> range

# what does the function do?

**Examples**

Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces

end

**Definition**

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

fun function name (variable(s)):

function name variable(s)

what the function does with those variable(s)

end

Directions:

**Contract and Purpose Statement**

Every contract has three parts…

# function name :: domain -> range

# what does the function do?

**Examples**

Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces

end

**Definition**

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

fun function name (variable(s)):

function name variable(s)

what the function does with those variable(s)

end
Word Problem: double-radius

Directions: 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 and Purpose Statement
Every contract has three parts…
# function name :: domain -> range
# what does the function do?

Examples
Write some examples, then circle and label what changes…
examples:

function name (input(s)) is what the function produces

end

Don't care
**Word Problem: double-radius**

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

**Contract and Purpose Statement**

Every contract has three parts…

```
# ____________________ ; ____________________ , ____________________   ->   ____________________
    function name                                  domain                        range
# ____________________
    what does the function do?                      
```

**Examples**

Write some examples, then circle and label what changes…

```
examples:  

function name ( _________ ) is _________  
what the function produces  

function name ( _________ ) is _________  
what the function produces  

end
```

Don't care
Word Problem: next-position

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

Contract and Purpose Statement

Every contract has three parts…

# function name :: domain -> range

# what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces

function name (input(s)) is what the function produces

end

Don't care
Data Structure

```haskell
data CakeType:
    | cake(____________________________________________________
    | ______________________________________________________
    | ______________________________________________________
end

to make an instance of this structure, i would write:

cake1 = ________________________________________________
cake2 = ________________________________________________

to access the fields of cake2, i would write:

____________________________________________________
____________________________________________________
____________________________________________________
```
Word Problem: taller-than

Directions: Write a function called `taller-than`, which consumes two CakeTypes, and produces true if the number of layers in the first CakeType is greater than the number of layers in the second.

Contract and Purpose Statement

Every contract has three parts…

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

# what does the function do?
```

Examples

Write some examples, then circle and label what changes…

```
examples:

function name (input(s)) is what the function produces

function name (input(s)) is what the function produces

end
```

Don't care
Word Problem: will-melt

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

Contract and Purpose Statement

Every contract has three parts…

Contract:

Examples

Write some examples, then circle and label what changes…

Examples:

Don’t care
Word Problem: draw-state

Write a function called `draw-state`, which takes in a SunsetState and returns an image in which the sun (a circle) appears at the position given in the SunsetState. The sun should be behind the horizon (the ground) once it is low in the sky.

**Contract and Purpose Statement**

`draw-state :: ___________________________ -> Image`

# ___________________________

**Write an expression for each piece of your final image**

- SUN =
- GROUND =
- SKY =

**Write the draw-state function, using put-image to combine your pieces**

```
fun (____________________) ________________________________ ;

_____________________________________________________________

_____________________________________________________________

_____________________________________________________________

end
```
Word Problem: next-state-tick

Directions: Write a function called next-state-tick, which takes in a SunsetState and returns a SunsetState in which the new x-coordinate is 8 pixels larger than in the given SunsetState and the y-coordinate is 4 pixels smaller than in the given SunsetState.

Contract and Purpose Statement
Every contract has three parts...

# function name :: domain -> range

# what does the function do?

Examples
Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces

function name (input(s)) is what the function produces

end

Don't care
Identifying Animation Data Worksheet: Sunset

Draw a sketch for three distinct moments of the animation

Sketch A

Sketch B

Sketch C

What things are changing?

<table>
<thead>
<tr>
<th>Thing</th>
<th>Describe how it changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What fields do you need to represent the things that change?

<table>
<thead>
<tr>
<th>Field name (dangerX, score, playerIMG ...)</th>
<th>Datatype (Number, String, Image, Boolean ...)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Animation Data Worksheet

**Draw a sketch for three distinct moments of the animation**

<table>
<thead>
<tr>
<th>Sketch A</th>
<th>Sketch B</th>
<th>Sketch C</th>
</tr>
</thead>
</table>

## What things are changing?

<table>
<thead>
<tr>
<th>Thing</th>
<th>Describe how it changes</th>
</tr>
</thead>
</table>

## What fields do you need to represent the things that change?

<table>
<thead>
<tr>
<th>Field name (dangerX, score, playerIMG …)</th>
<th>Datatype (Number, String, Image, Boolean …)</th>
</tr>
</thead>
</table>

## Make a To-Do List, and check off each as “Done” when you finish each one.

<table>
<thead>
<tr>
<th>Component</th>
<th>When is there work to be done?</th>
<th>To-Do</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Structure</td>
<td>If any new field(s) were added, changed, or removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>draw-state</td>
<td>If something is displayed in a new way or position</td>
<td></td>
<td></td>
</tr>
<tr>
<td>next-state-tick</td>
<td>If the Data Structure changed, or the animation happens automatically</td>
<td></td>
<td></td>
</tr>
<tr>
<td>next-state-key</td>
<td>If the Data Structure changed, or a keypress triggers the animation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reactor</td>
<td>If either next-state function is new</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Define the Data Structure

# a State is 

data State:
| ( |
| | |
| | |
| | |
| | |
| | |
|
end

Make a sample instance for each sketch from the previous page

_______ = 

_______ = 

_______ = 

Write an example for one of the functions on the previous page

__________________________
__________________________
__________________________
__________________________
__________________________
__________________________
Collisions
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:

\[(4 - 0)^2 + (2 - 5)^2\]

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

```
(num-sqr (- 4 0) (+ (- 2 5) (- 2 5)))
```

Convert it to Pyret code.
Directions: Write a function distance, which takes FOUR inputs: (1) px: The x-coordinate of the player, (2) py: The y-coordinate of the player, (3) cx: The x-coordinate of another game character, (4) cy: The y-coordinate of another game character. It should return the distance between the two, using the Distance formula: Distance² = (px - cx)² + (py - cy)²

Contract and Purpose Statement
Every contract has three parts…
# ___________________ :: ___________________ -> ___________________  
# ___________________  

what does the function do?

Examples
Write some examples, then circle and label what changes…

examples:

[ ]

end

Don't care
Word Problem: is-collision

Directions: Write a function is-collision, which takes FOUR inputs: (1) px: The x-coordinate of the player, (2) py: The y-coordinate of the player, (3) cx: The x-coordinate of another game character, (4) 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 and Purpose Statement

Every contract has three parts...

# ___________________ :: ___________________ --> ___________________

# ___________________ : ___________________

what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

(__________) is ___________________

(__________) is ___________________

end

Don't care
Contracts

Contracts tell us how to use a function. For example: \( \text{num-sqr :: (n :: Number) -> Number} \) tells us that the name of the function is \( \text{num-sqr} \), it takes one input (a Number), and it evaluates to a Number. From the contract, we know \( \text{num-sqr}(4) \) will evaluate to a Number.

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td># triangle</td>
<td>( (\text{side-length :: Number, style :: String, color :: String}) )</td>
<td>( \rightarrow ) Image</td>
</tr>
<tr>
<td># circle</td>
<td>( (\text{radius :: Number, style :: String, color :: String}) )</td>
<td>( \rightarrow ) Image</td>
</tr>
<tr>
<td># star</td>
<td>( (\text{radius :: Number, style :: String, color :: String}) )</td>
<td>( \rightarrow ) Image</td>
</tr>
<tr>
<td># rectangle</td>
<td>( (\text{width :: Num, height :: Num, style :: Str, color :: Str}) )</td>
<td>( \rightarrow ) Image</td>
</tr>
<tr>
<td># ellipse</td>
<td>( (\text{width :: Num, height :: Num, style :: Str, color :: Str}) )</td>
<td>( \rightarrow ) Image</td>
</tr>
<tr>
<td># square</td>
<td>( (\text{size-length :: Number, style :: String, color :: String}) )</td>
<td>( \rightarrow ) Image</td>
</tr>
<tr>
<td># text</td>
<td>( (\text{str :: String, size :: Number, color :: String}) )</td>
<td>( \rightarrow ) Image</td>
</tr>
<tr>
<td># overlay</td>
<td>( (\text{img1 :: Image, img2 :: Image}) )</td>
<td>( \rightarrow ) Image</td>
</tr>
<tr>
<td># beside</td>
<td>( (\text{img1 :: Image, img2 :: Image}) )</td>
<td>( \rightarrow ) Image</td>
</tr>
<tr>
<td># bitmap-url</td>
<td>( (\text{url :: String}) )</td>
<td>( \rightarrow ) Image</td>
</tr>
</tbody>
</table>
Contracts

Contracts tell us how to use a function. For example, `num-sqr :: (n :: Number) -> Number` tells us that the name of the function is `num-sqr`, it takes one input (a `Number`), and it evaluates to a `Number`. From the contract, we know `num-sqr(4)` will evaluate to a `Number`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td># above</td>
<td>:: (img1 :: Image, img2 :: Image)</td>
<td>-&gt;</td>
</tr>
<tr>
<td># put-image</td>
<td>:: (img1 :: Image, x :: Number, y :: Number, img2 :: Image)</td>
<td>-&gt;</td>
</tr>
<tr>
<td># rotate</td>
<td>:: (degree :: Number, img :: Image)</td>
<td>-&gt;</td>
</tr>
<tr>
<td># scale</td>
<td>:: (factor :: Number, img :: Image)</td>
<td>-&gt;</td>
</tr>
<tr>
<td># string-repeat</td>
<td>:: (text :: String, repeat :: Number)</td>
<td>-&gt;</td>
</tr>
<tr>
<td># string-contains</td>
<td>:: (text :: String, search-for :: String)</td>
<td>-&gt;</td>
</tr>
<tr>
<td># num-sqr</td>
<td>:: (n :: Number)</td>
<td>-&gt;</td>
</tr>
<tr>
<td># num-sqrt</td>
<td>:: (n :: Number)</td>
<td>-&gt;</td>
</tr>
<tr>
<td># num-min</td>
<td>:: (a :: Number, b:: Number)</td>
<td>-&gt;</td>
</tr>
<tr>
<td># num-max</td>
<td>:: (a :: Number, b:: Number)</td>
<td>-&gt;</td>
</tr>
</tbody>
</table>
Contracts

Contracts tell us how to use a function. For example: num-sqr :: (n :: Number) -> Number tells us that the name of the function is num-sqr, it takes one input (a Number), and it evaluates to a Number. From the contract, we know num-sqr(4) will evaluate to a Number.

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td># string-equal</td>
<td>:: (str1 :: String, str2 :: String)</td>
<td>-&gt; Boolean</td>
</tr>
<tr>
<td># and</td>
<td>:: (test1 :: Boolean, test2 :: Boolean)</td>
<td>-&gt; Boolean</td>
</tr>
<tr>
<td># or</td>
<td>:: (test1 :: Boolean, test2 :: Boolean)</td>
<td>-&gt; Boolean</td>
</tr>
</tbody>
</table>