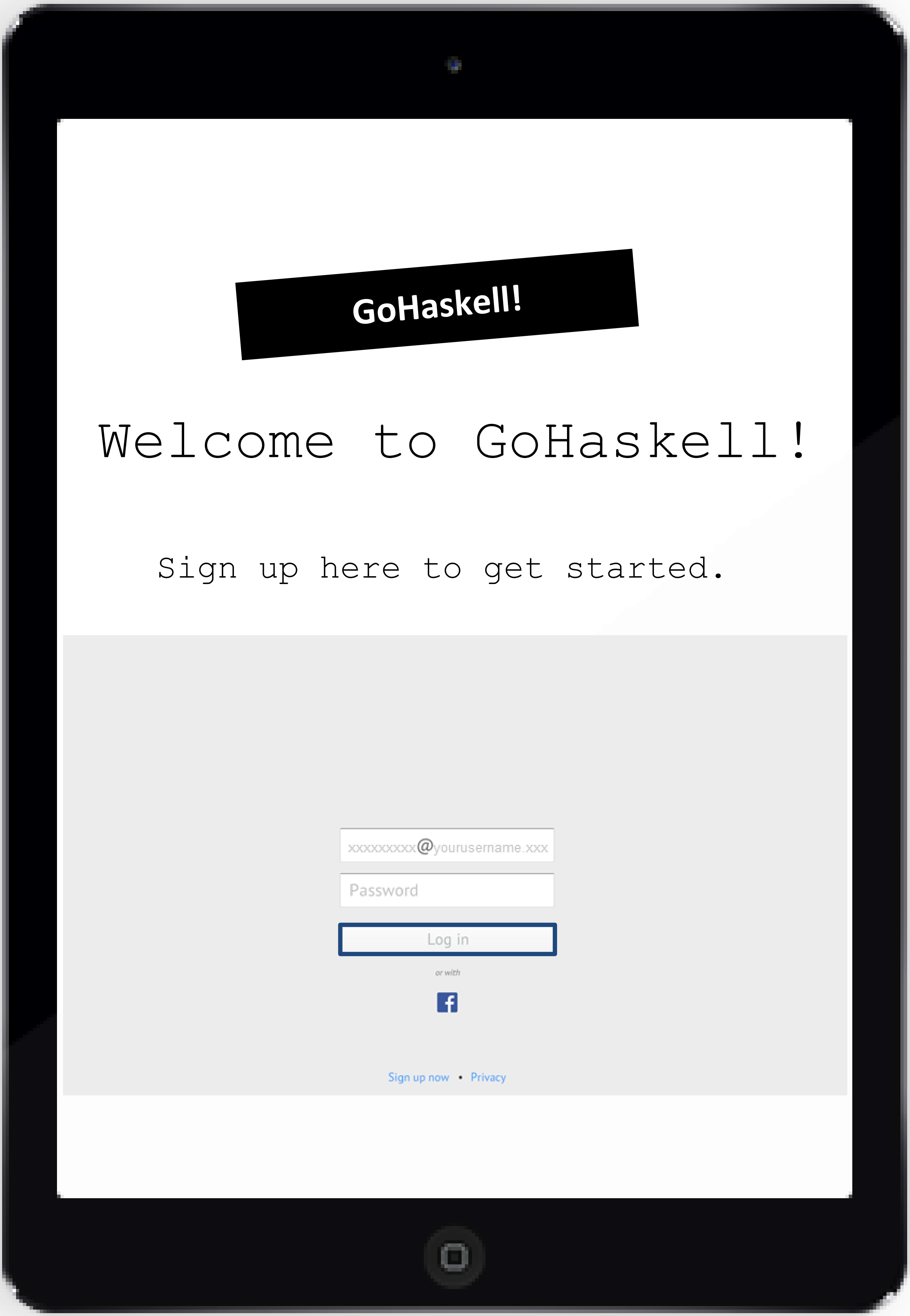
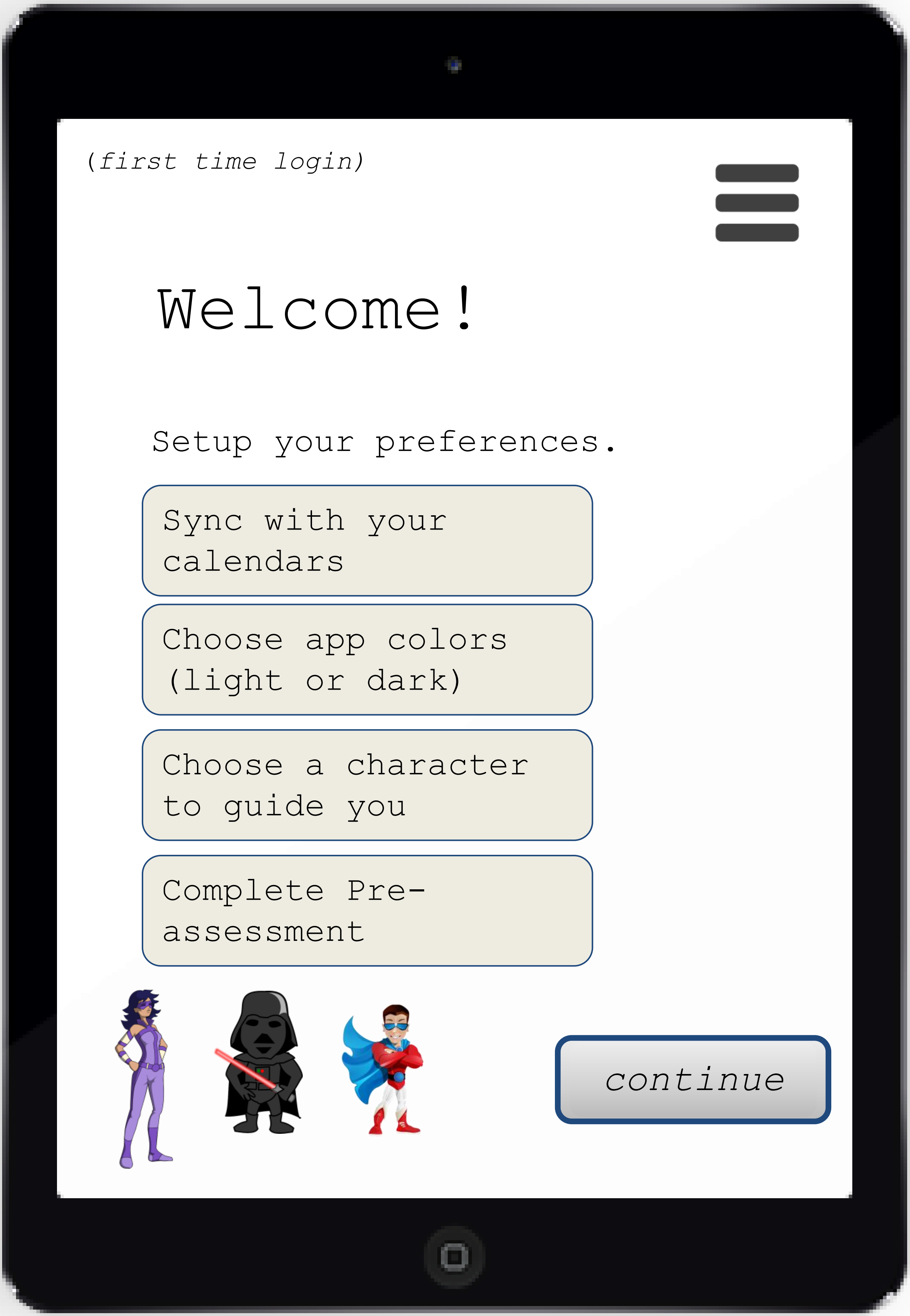


GoHaskell!

Welcome to GoHaskell!

Sign up here to get started.



*(first time login)*

☰

Welcome!

Setup your preferences.

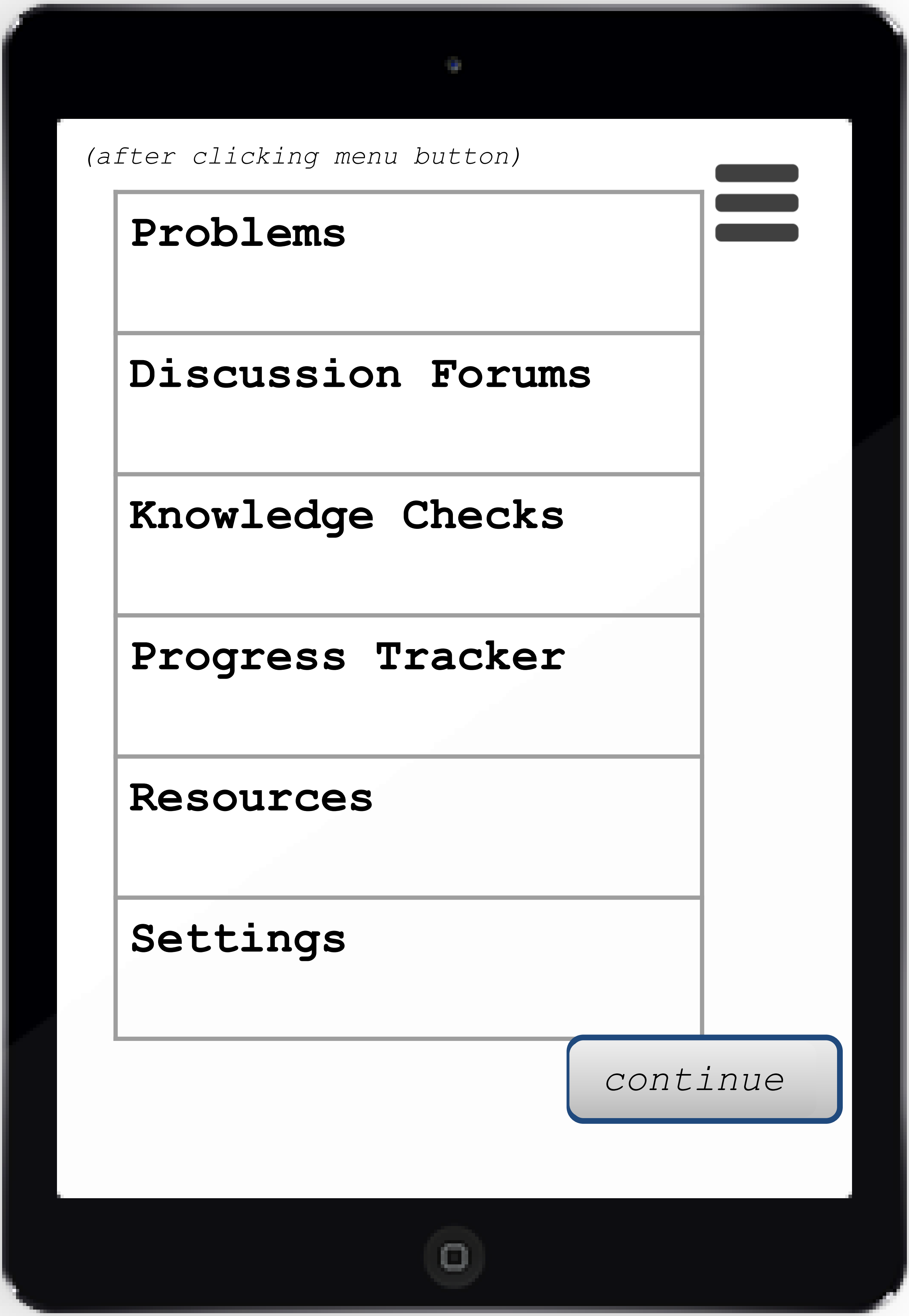
Sync with your calendars

Choose app colors (light or dark)

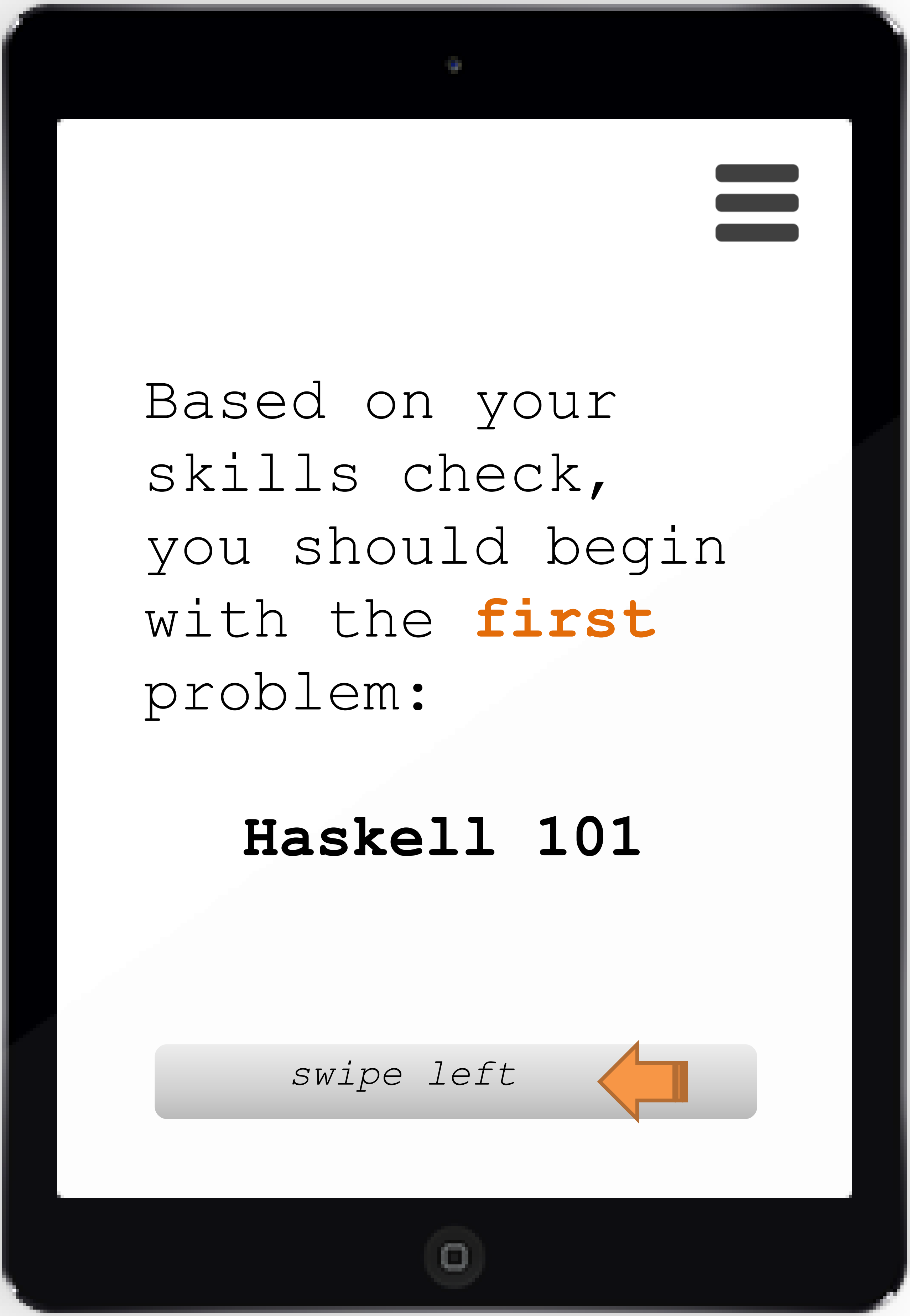
Choose a character to guide you

Complete Pre-assessment

*continue*



(after clicking menu button)

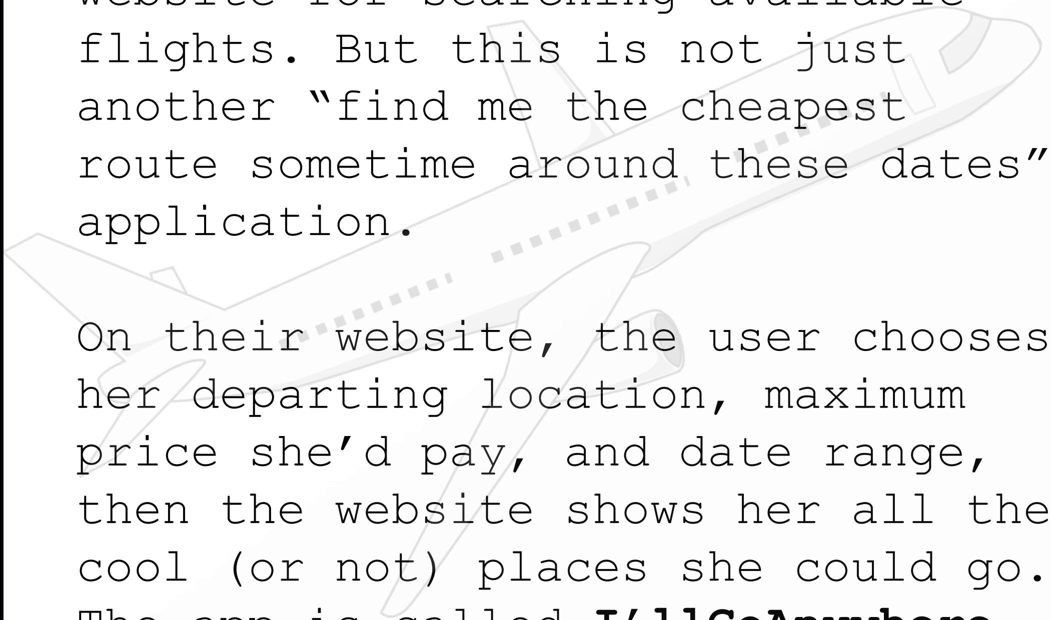


Based on your skills check, you should begin with the **first** problem:

**Haskell 101**

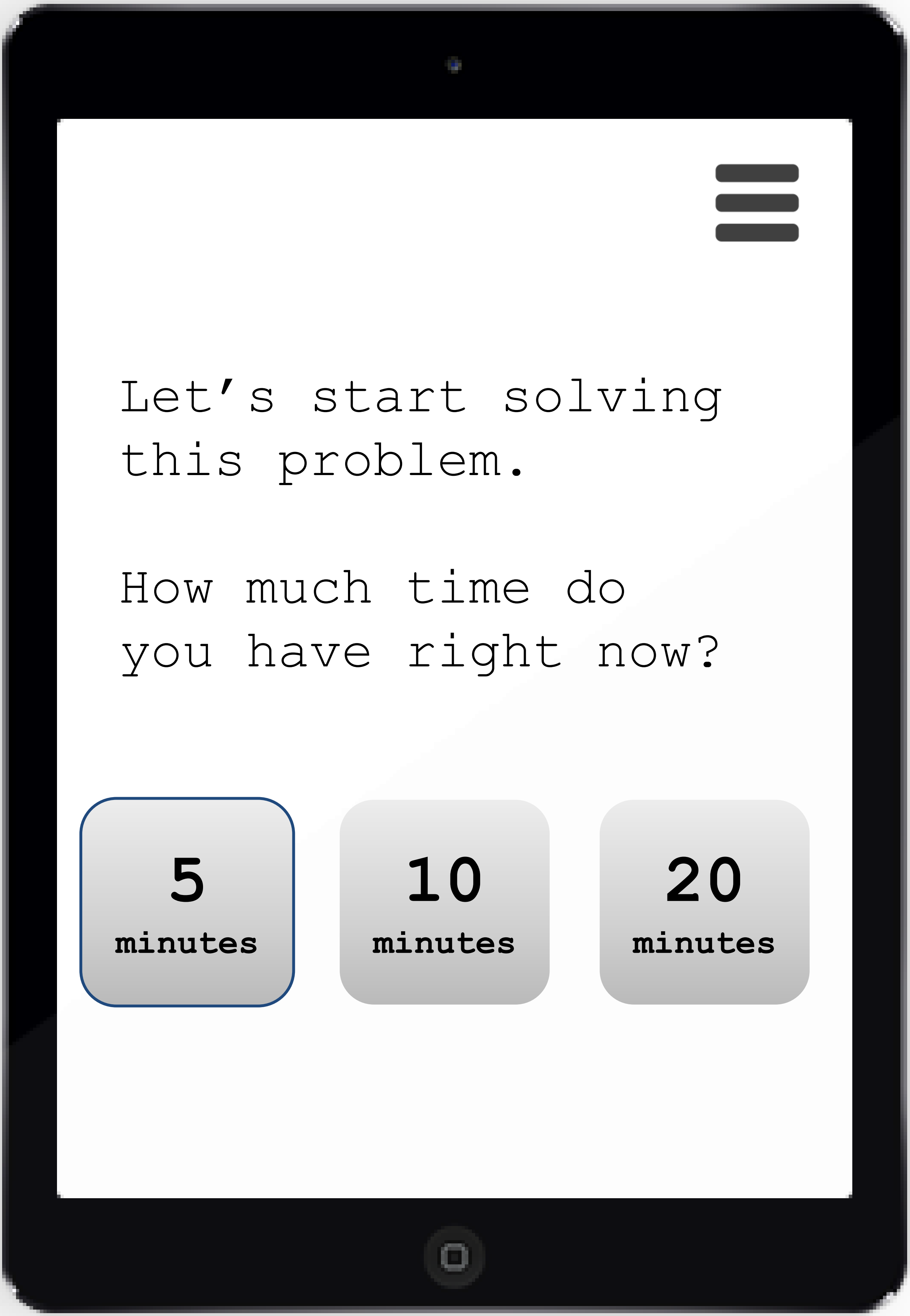
Haskell 101 > Problem Description

You've just taken a new job at a start-up that wants to develop a website for searching available flights. But this is not just another "find me the cheapest route sometime around these dates" application.



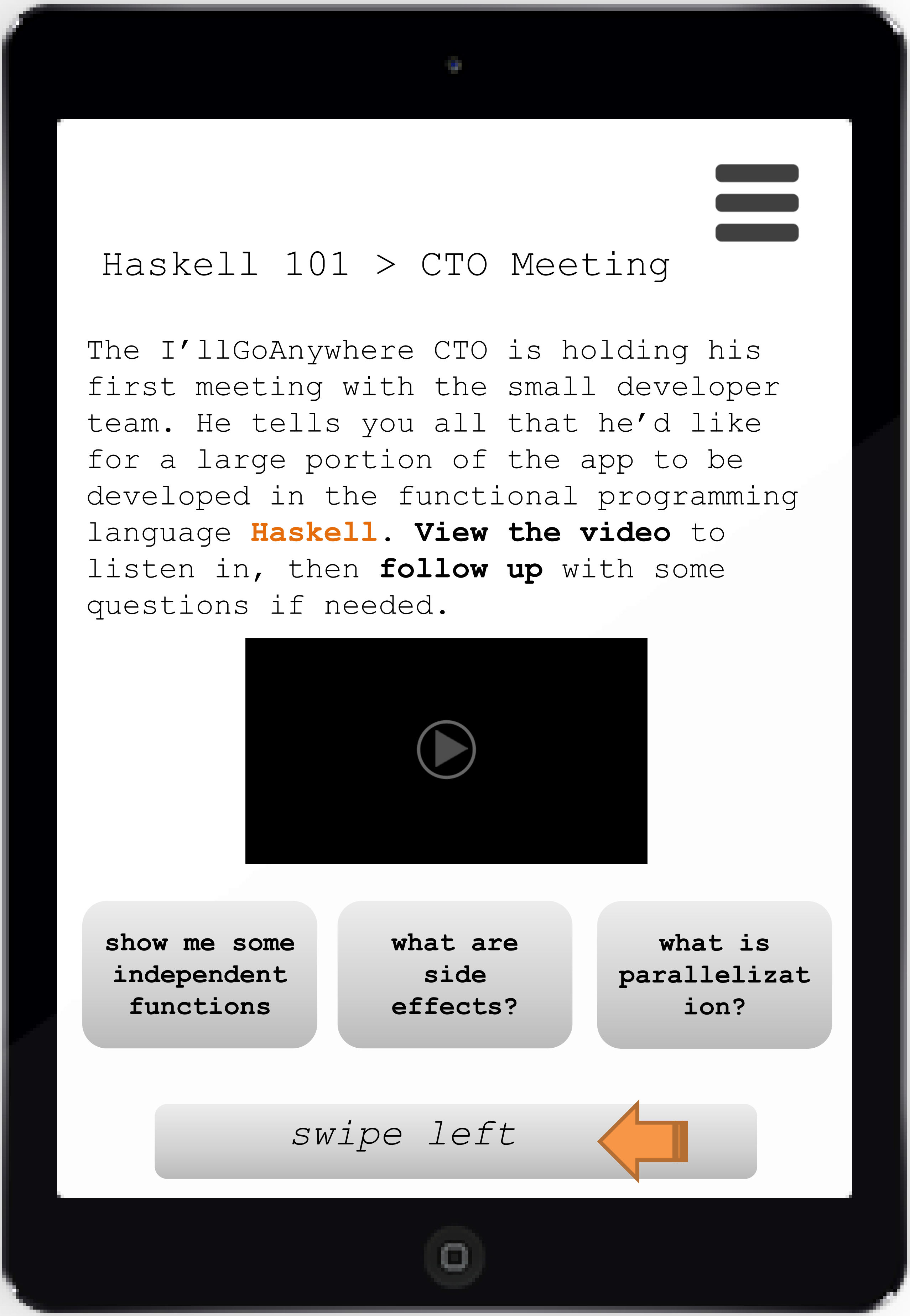
On their website, the user chooses her departing location, maximum price she'd pay, and date range, then the website shows her all the cool (or not) places she could go. The app is called **I'llGoAnywhere**.

*swipe left*



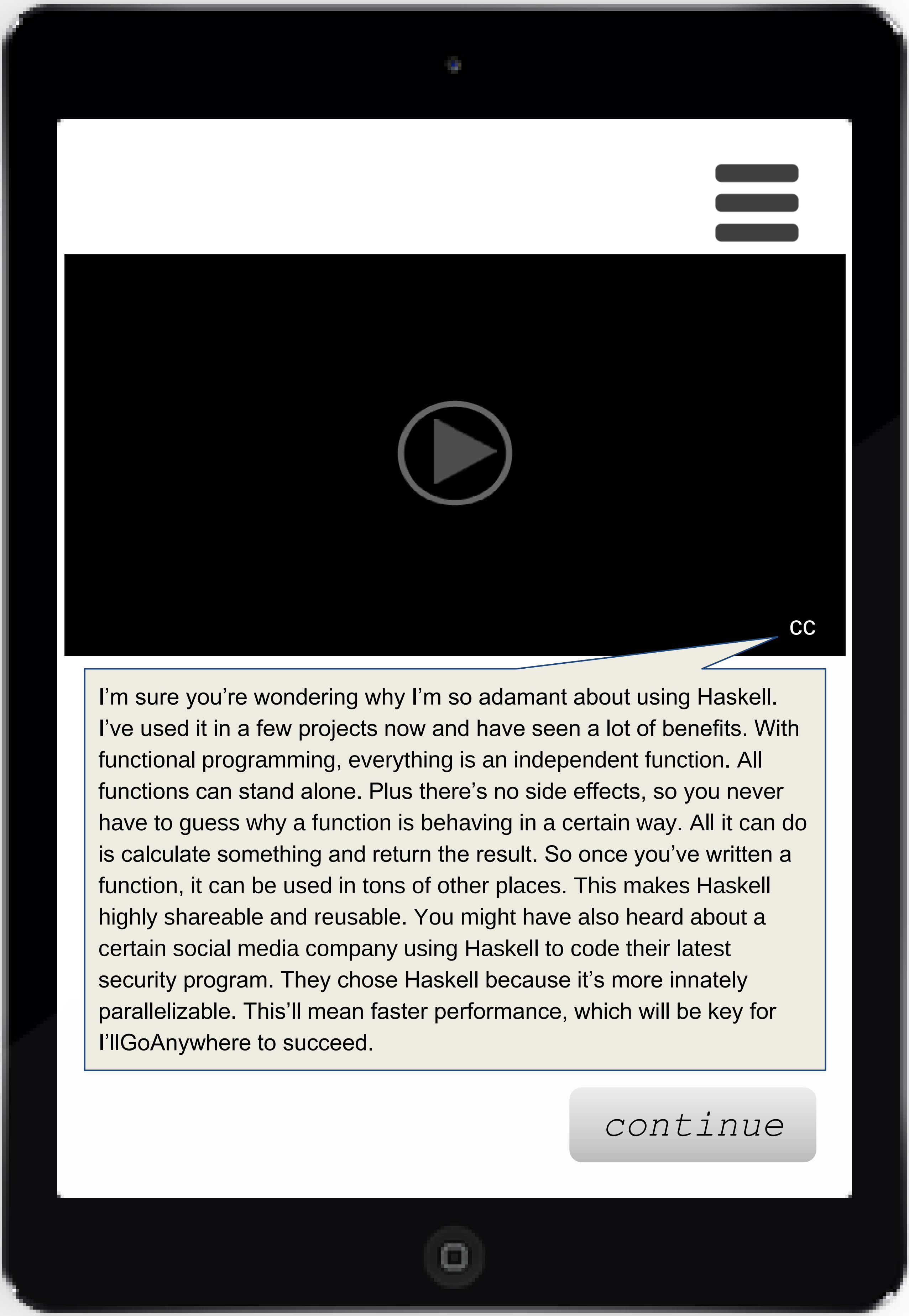
Let's start solving this problem.

How much time do you have right now?

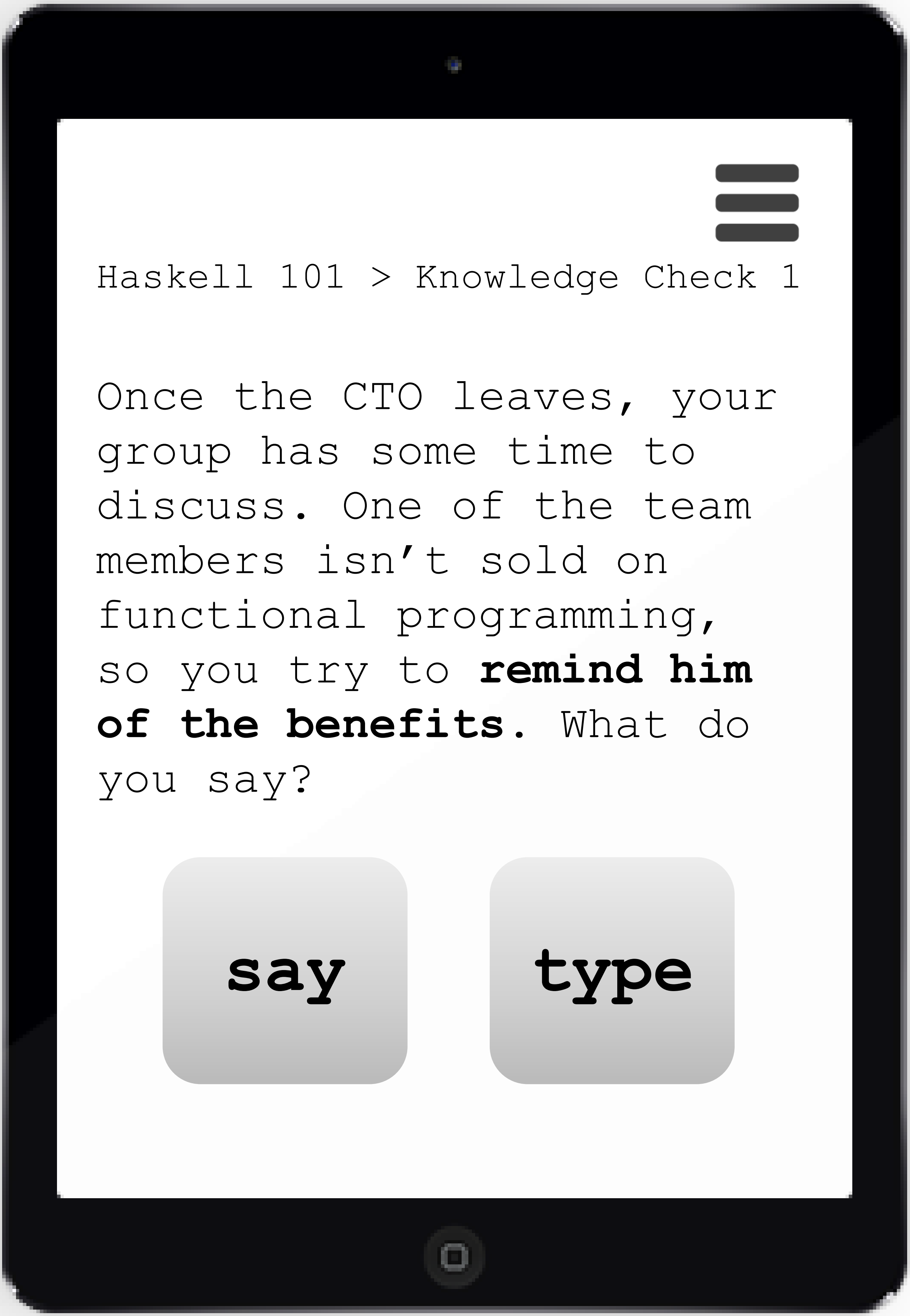


Haskell 101 > CTO Meeting

The I'llGoAnywhere CTO is holding his first meeting with the small developer team. He tells you all that he'd like for a large portion of the app to be developed in the functional programming language **Haskell**. **View the video** to listen in, then **follow up** with some questions if needed.



I’m sure you’re wondering why I’m so adamant about using Haskell. I’ve used it in a few projects now and have seen a lot of benefits. With functional programming, everything is an independent function. All functions can stand alone. Plus there’s no side effects, so you never have to guess why a function is behaving in a certain way. All it can do is calculate something and return the result. So once you’ve written a function, it can be used in tons of other places. This makes Haskell highly shareable and reusable. You might have also heard about a certain social media company using Haskell to code their latest security program. They chose Haskell because it’s more innately parallelizable. This’ll mean faster performance, which will be key for I’llGoAnywhere to succeed.

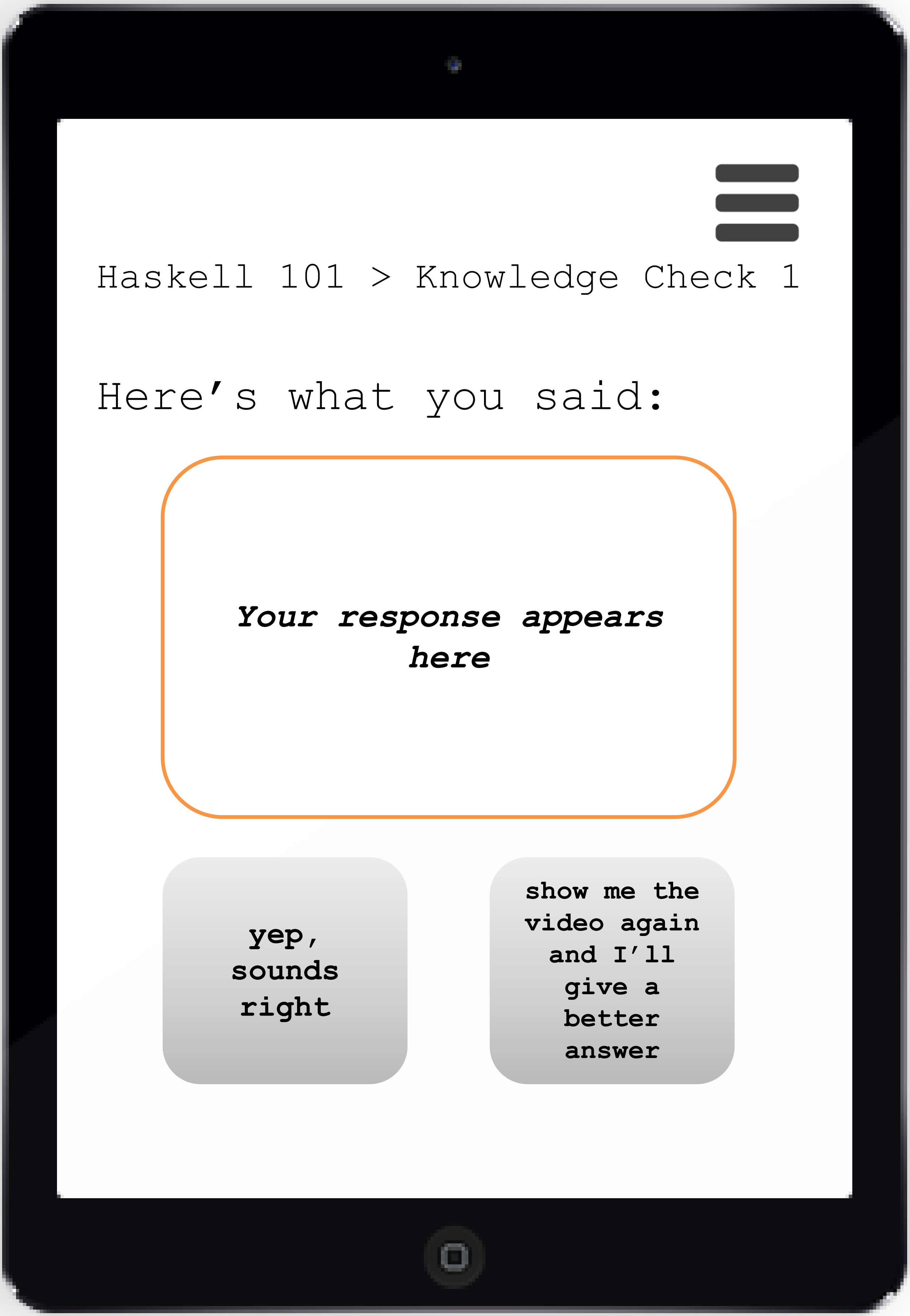


Haskell 101 > Knowledge Check 1

Once the CTO leaves, your group has some time to discuss. One of the team members isn’t sold on functional programming, so you try to **remind him of the benefits**. What do you say?

Recording

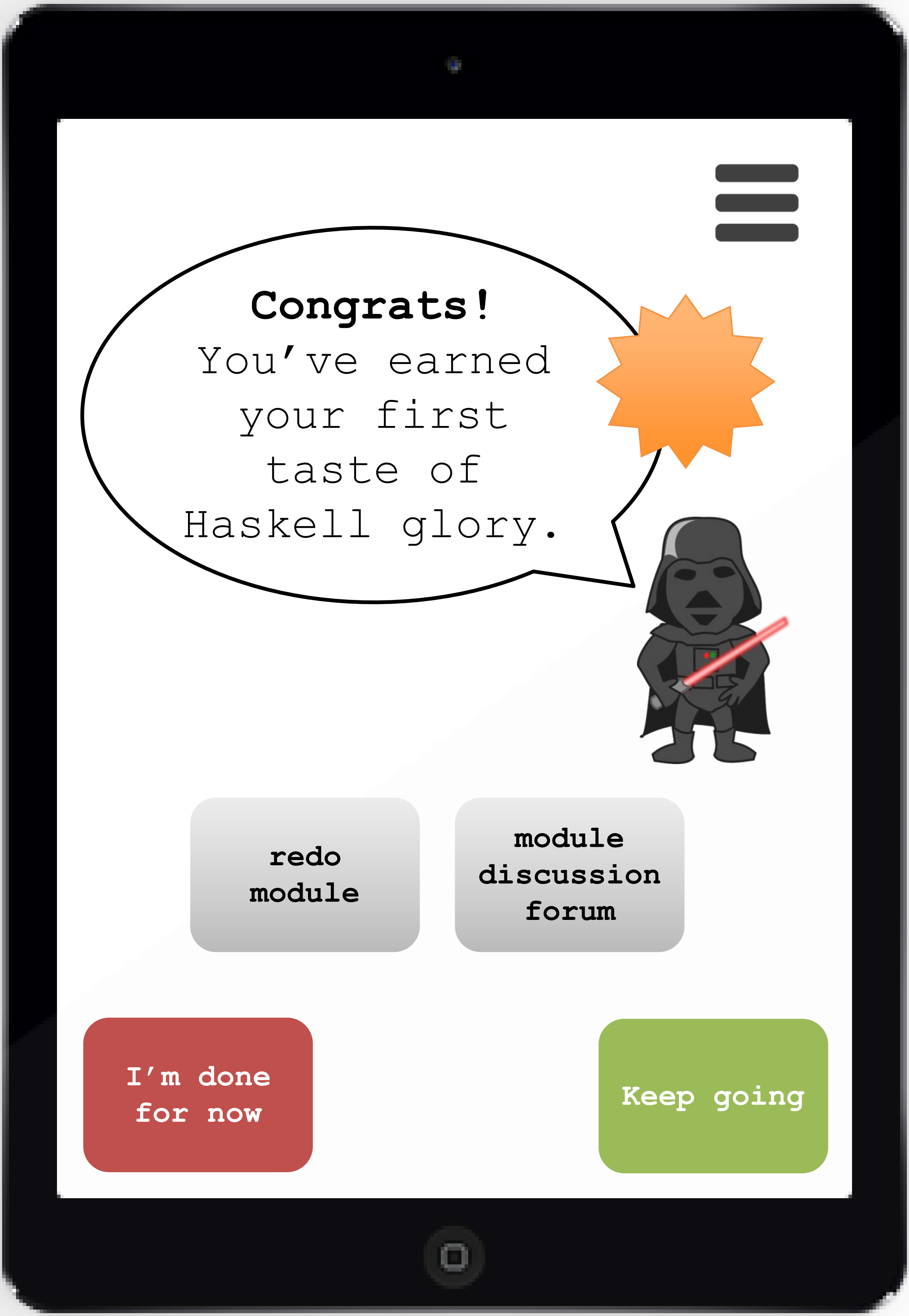
(Typed text appears here)



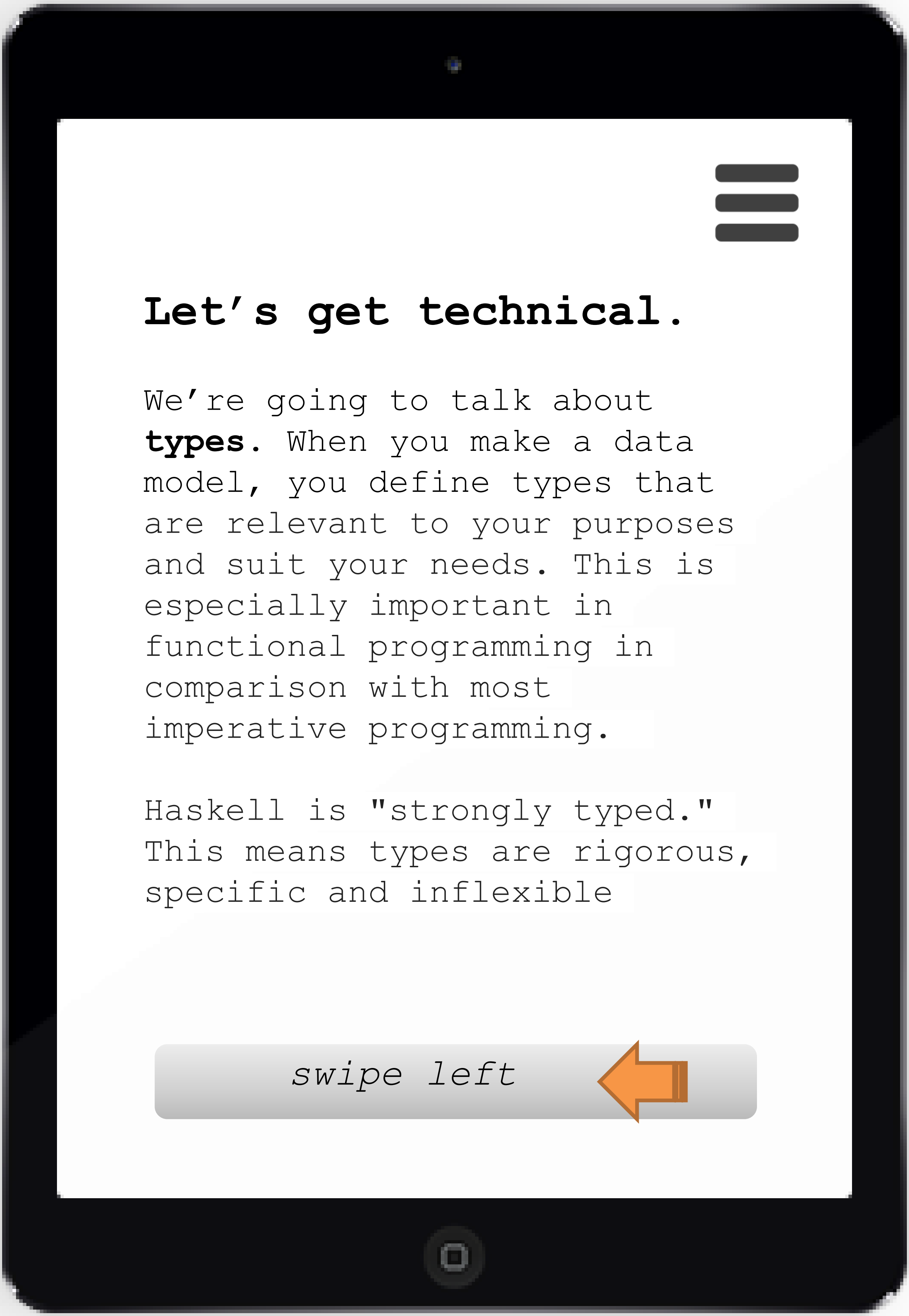
Haskell 101 > Knowledge Check 1

Here’s what you said:

***Your response appears here***



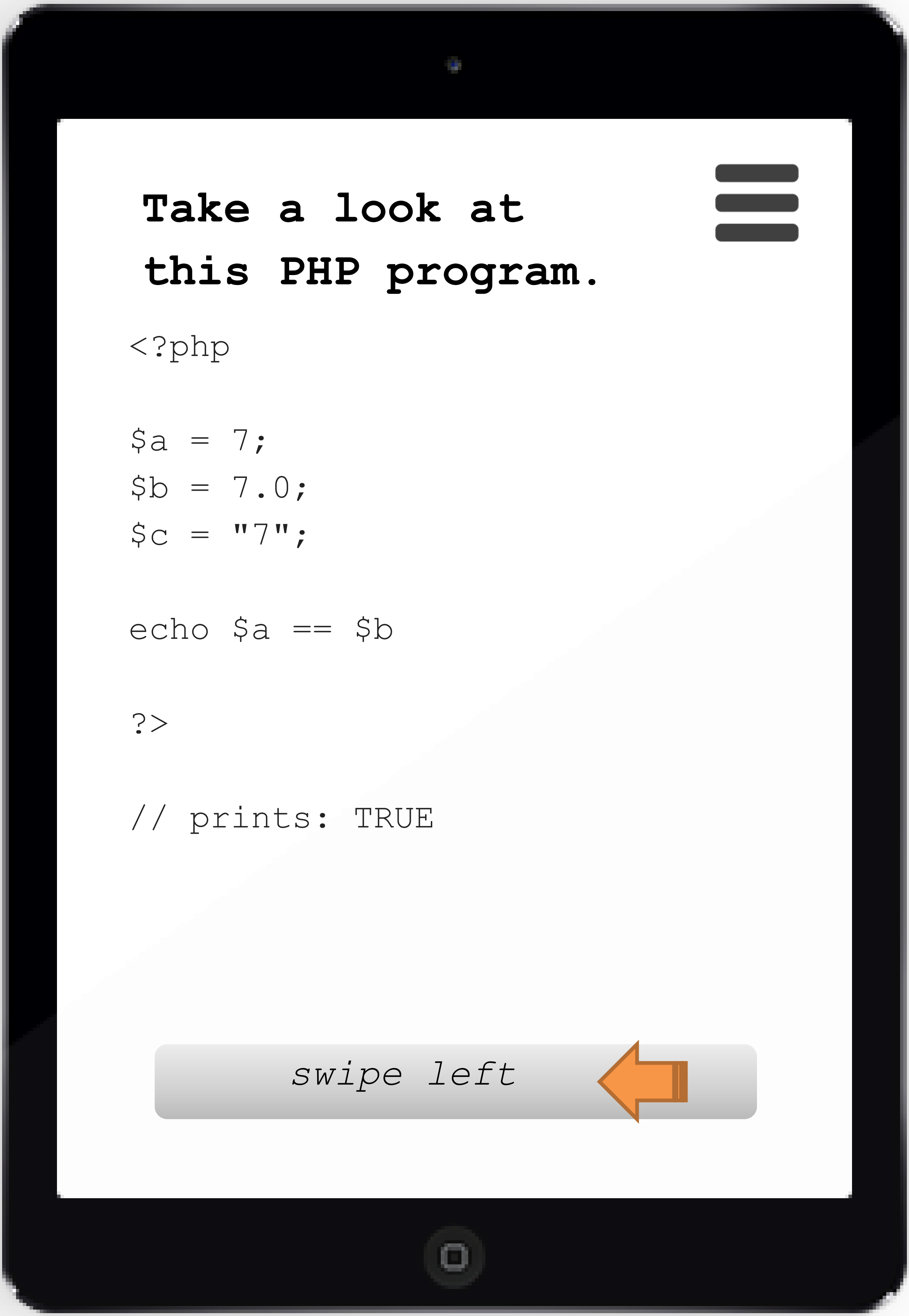
**Congrats!**
You've earned
your first
taste of
Haskell glory.



## Let's get technical.

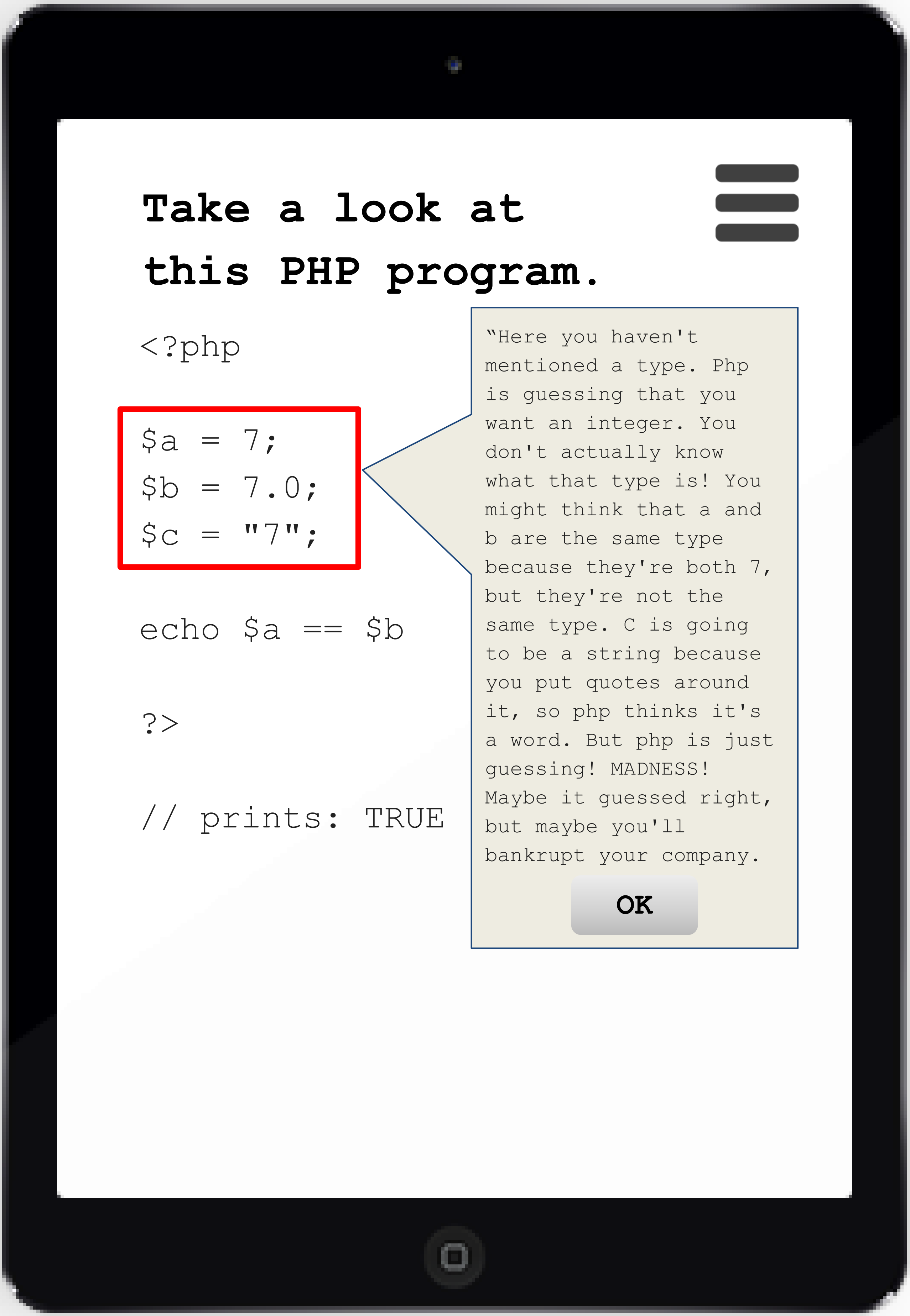
We're going to talk about **types.** When you make a data model, you define types that are relevant to your purposes and suit your needs. This is especially important in functional programming in comparison with most imperative programming.

Haskell is "strongly typed." This means types are rigorous, specific and inflexible



**Take a look at this PHP program.**

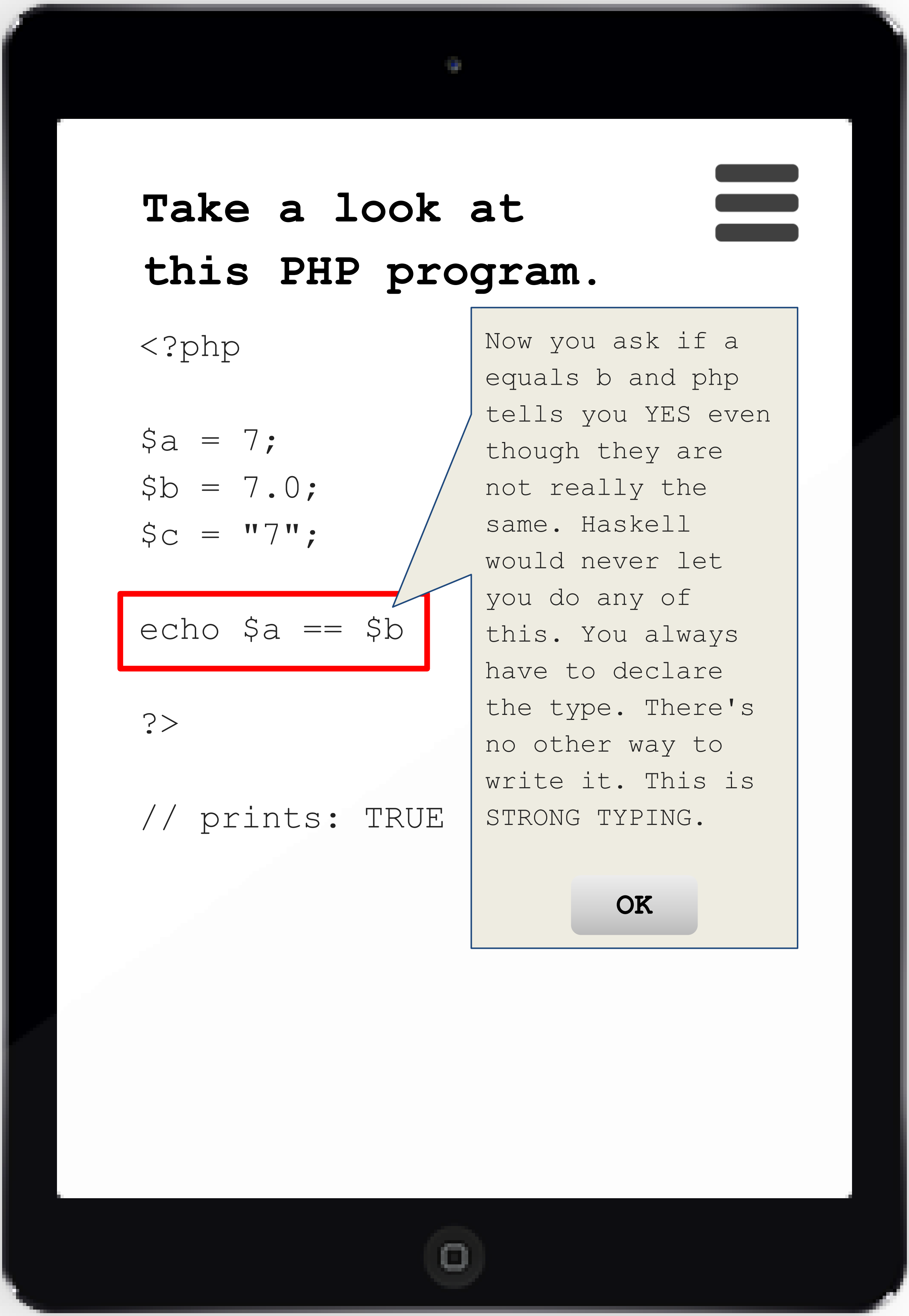
```
<?php

$a = 7;
$b = 7.0;
$c = "7";

echo $a == $b

?>

// prints: TRUE
```



# Take a look at this PHP program.

```
<?php

$a = 7;
$b = 7.0;
$c = "7";

echo $a == $b

?>

// prints: TRUE
```

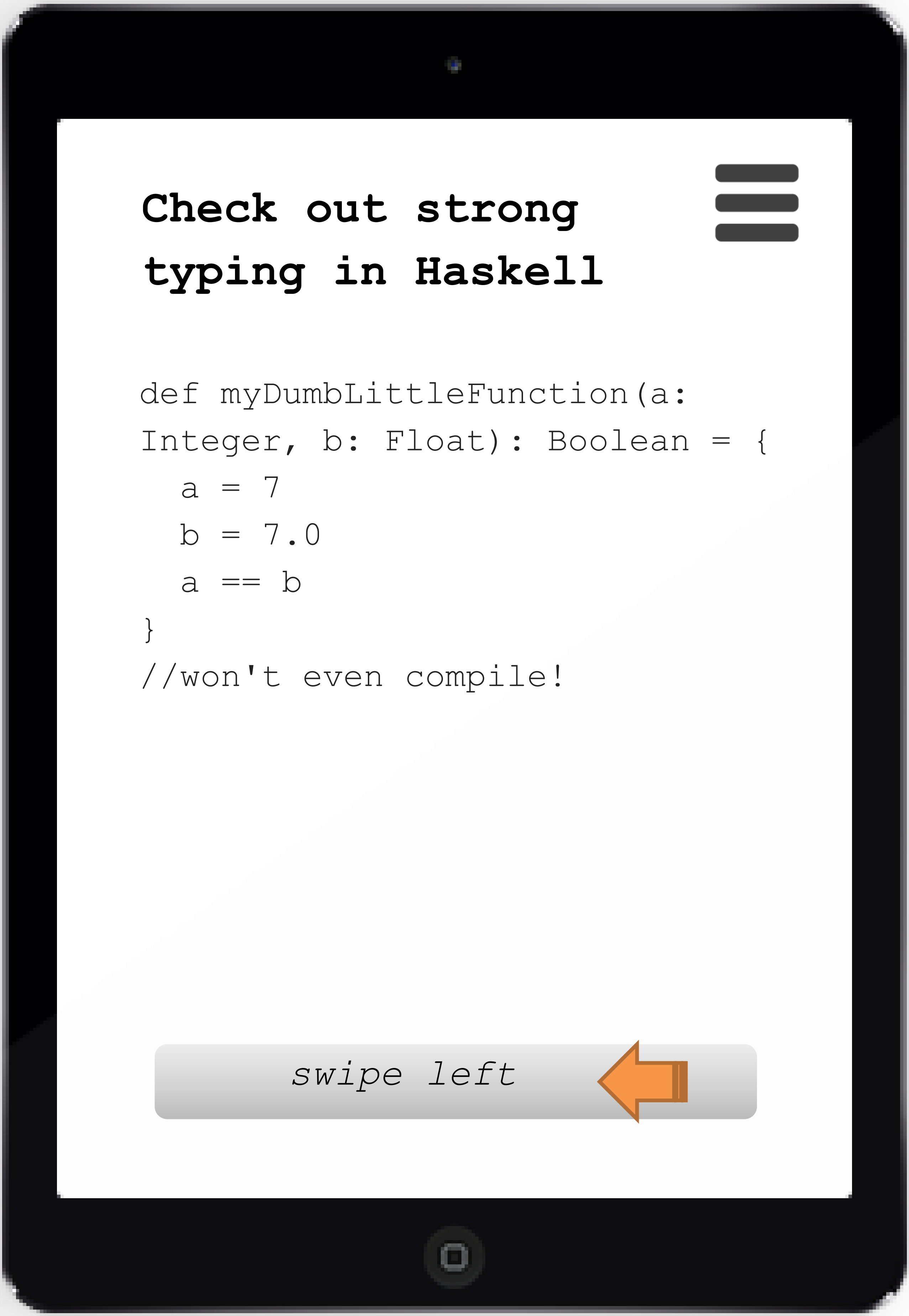
"Here you haven't mentioned a type. Php is guessing that you want an integer. You don't actually know what that type is! You might think that a and b are the same type because they're both 7, but they're not the same type. C is going to be a string because you put quotes around it, so php thinks it's a word. But php is just guessing! MADNESS! Maybe it guessed right, but maybe you'll bankrupt your company.



# Take a look at this PHP program.

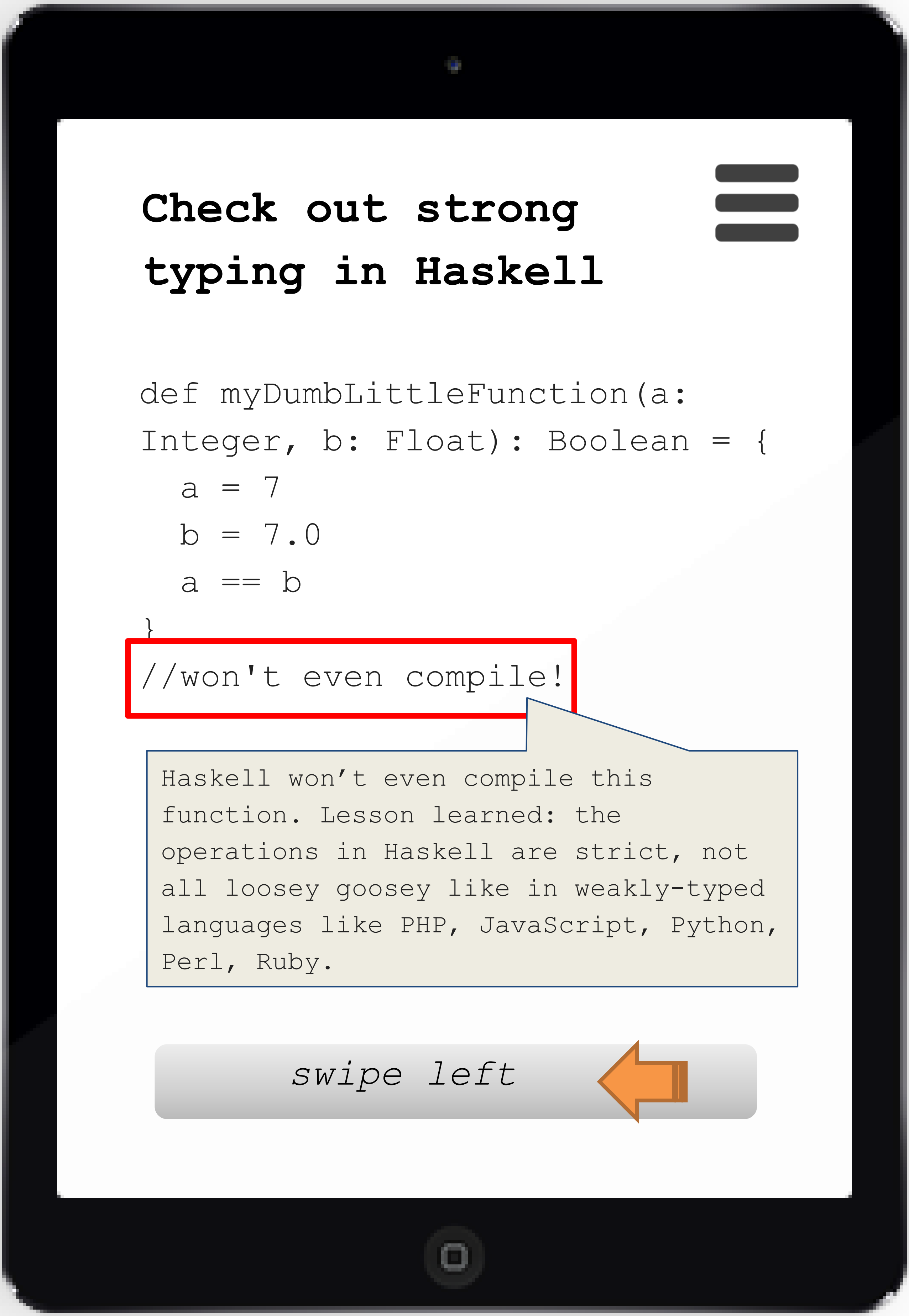
```
<?php

$a = 7;
$b = 7.0;
$c = "7";

echo $a == $b

?>

// prints: TRUE
```

Now you ask if a equals b and php tells you YES even though they are not really the same. Haskell would never let you do any of this. You always have to declare the type. There's no other way to write it. This is STRONG TYPING.



# Check out strong typing in Haskell

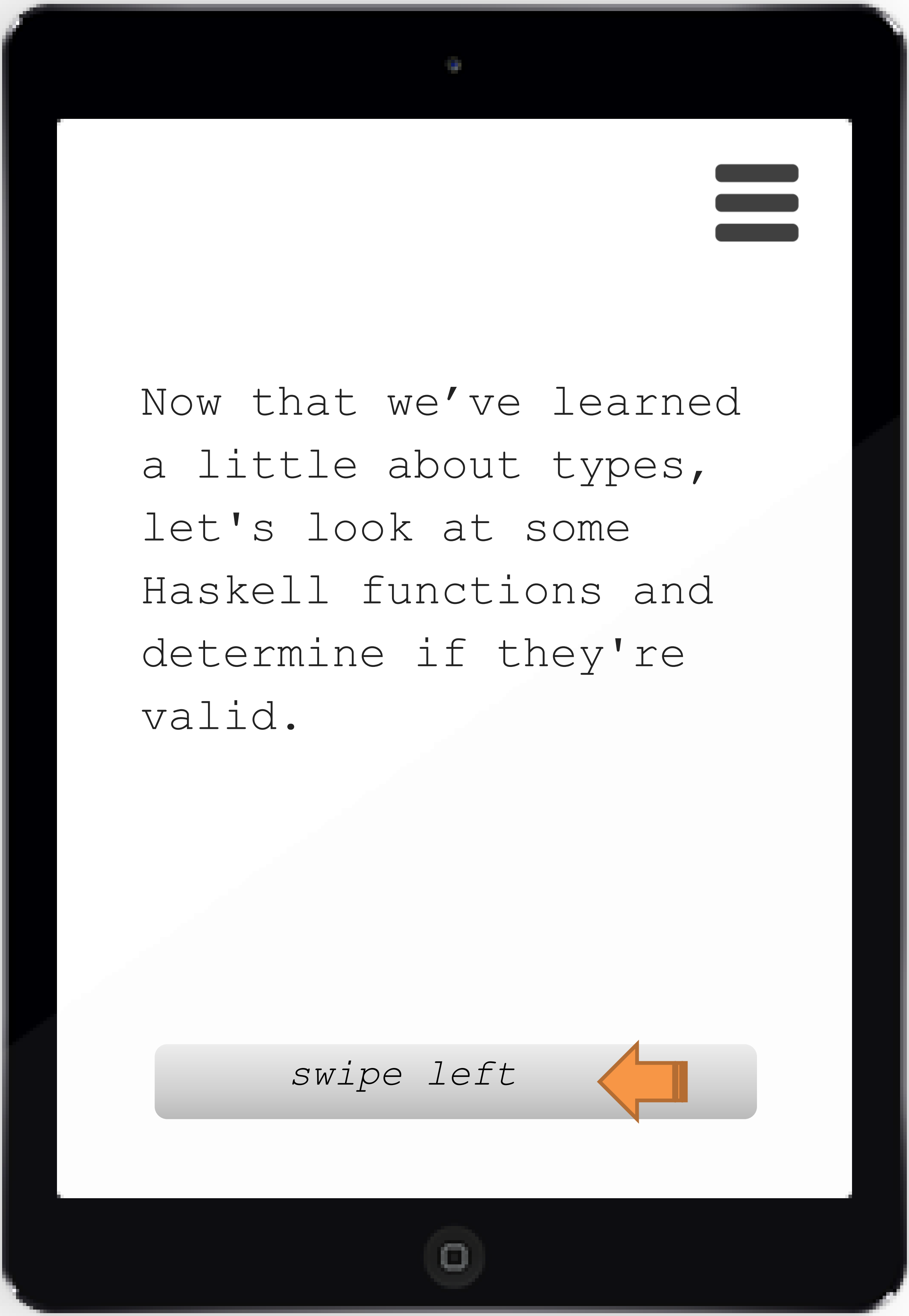
```
def myDumbLittleFunction(a:
Integer, b: Float): Boolean = {
  a = 7
  b = 7.0
  a == b
}
//won't even compile!
```



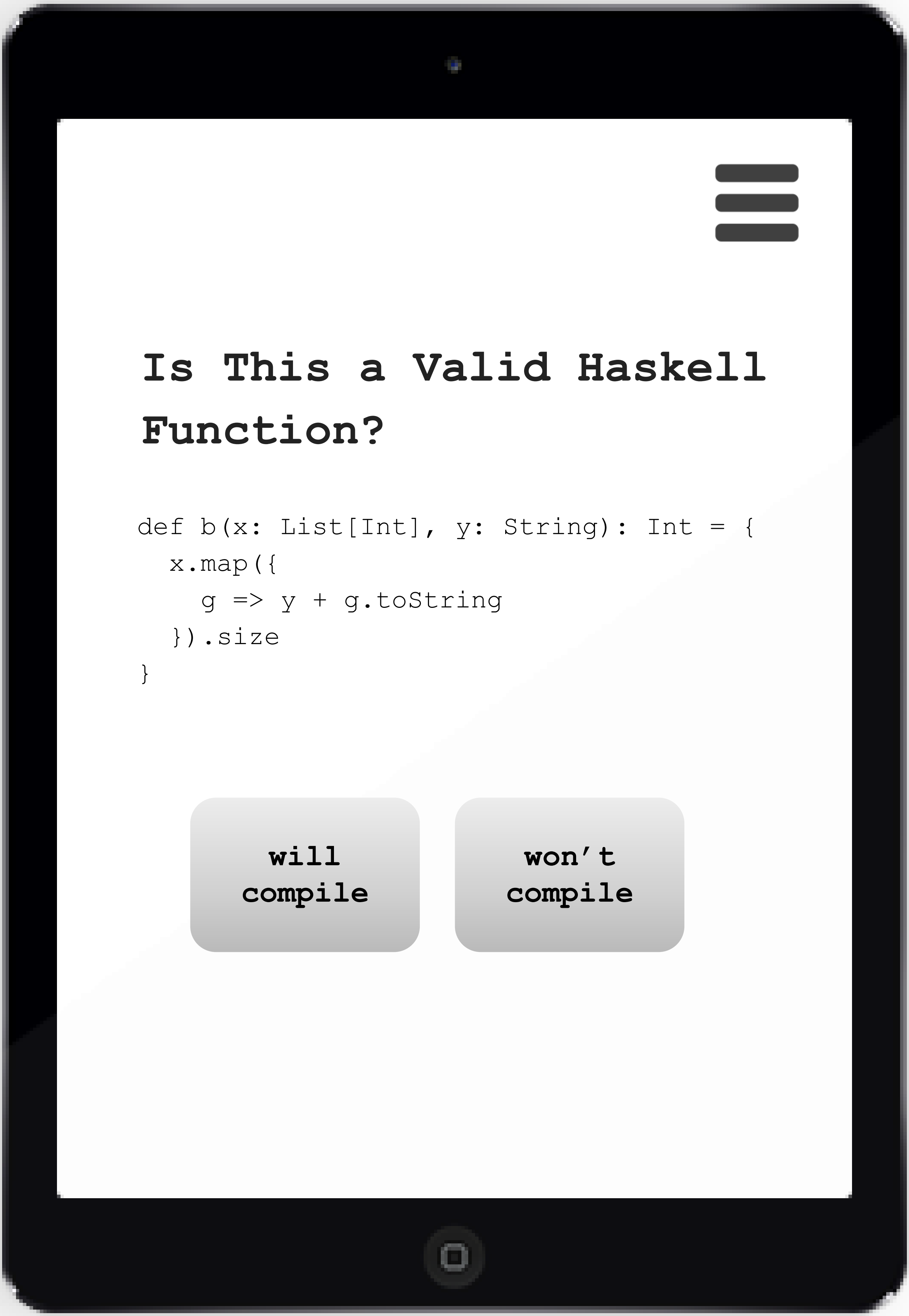
# Check out strong typing in Haskell

```
def myDumbLittleFunction(a:
Integer, b: Float): Boolean = {
   a = 7
   b = 7.0
   a == b
}
//won't even compile!
```

Haskell won’t even compile this function. Lesson learned: the operations in Haskell are strict, not all loosey goosey like in weakly-typed languages like PHP, JavaScript, Python, Perl, Ruby.

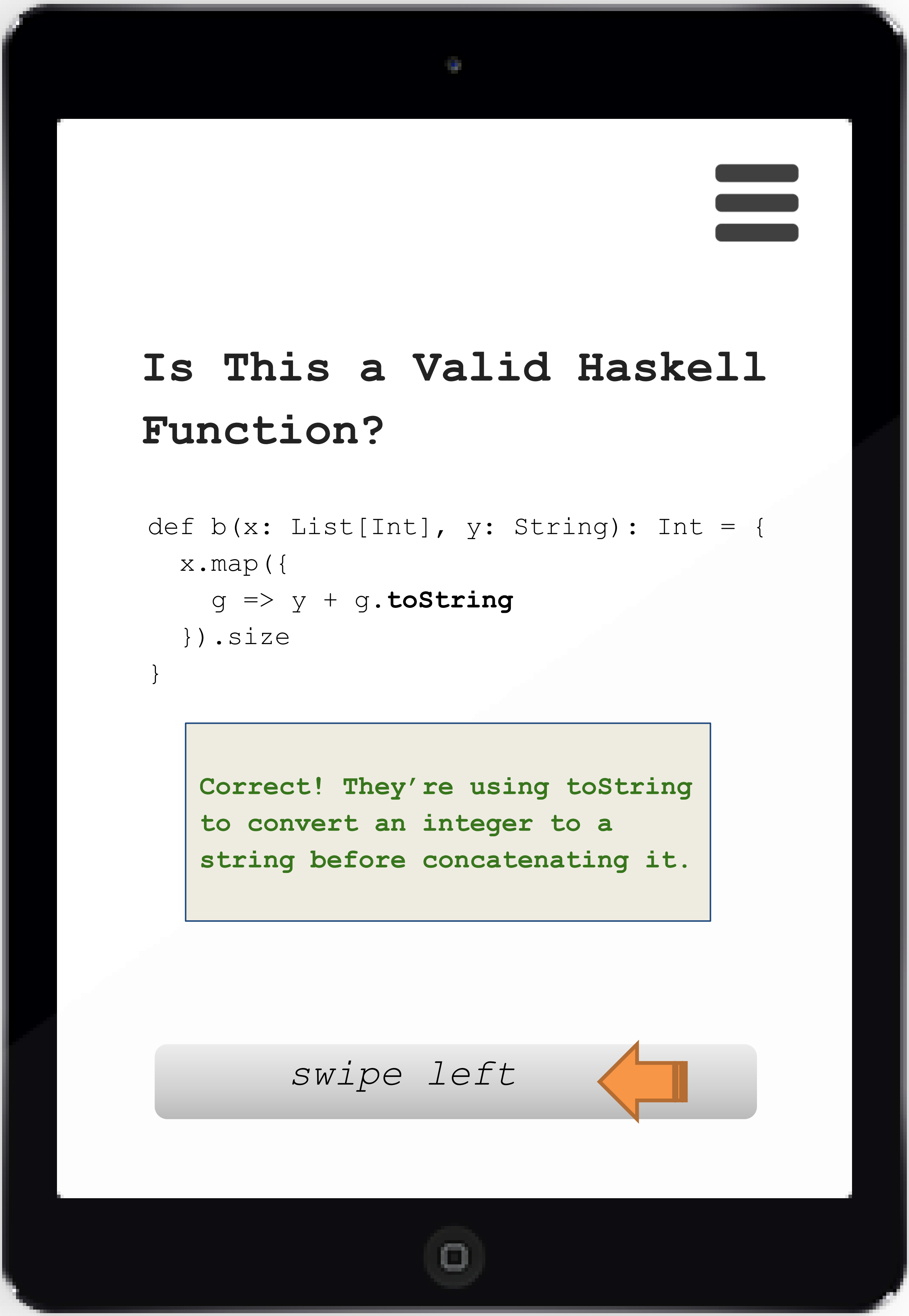


Now that we’ve learned a little about types, let's look at some Haskell functions and determine if they're valid.



# Is This a Valid Haskell Function?

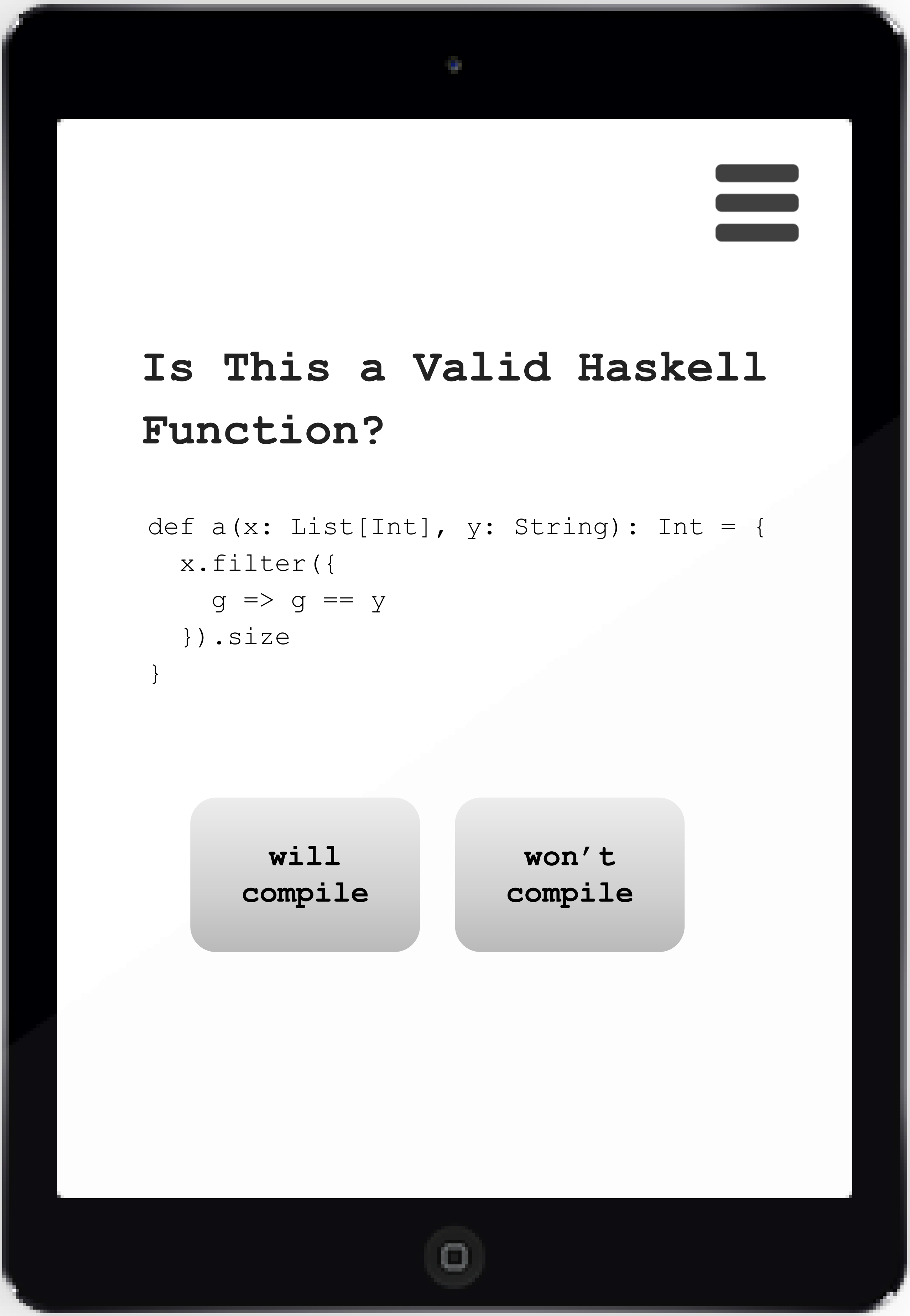
```
def b(x: List[Int], y: String): Int = {
  x.map({
    g => y + g.toString
  }).size
}
```



# Is This a Valid Haskell Function?

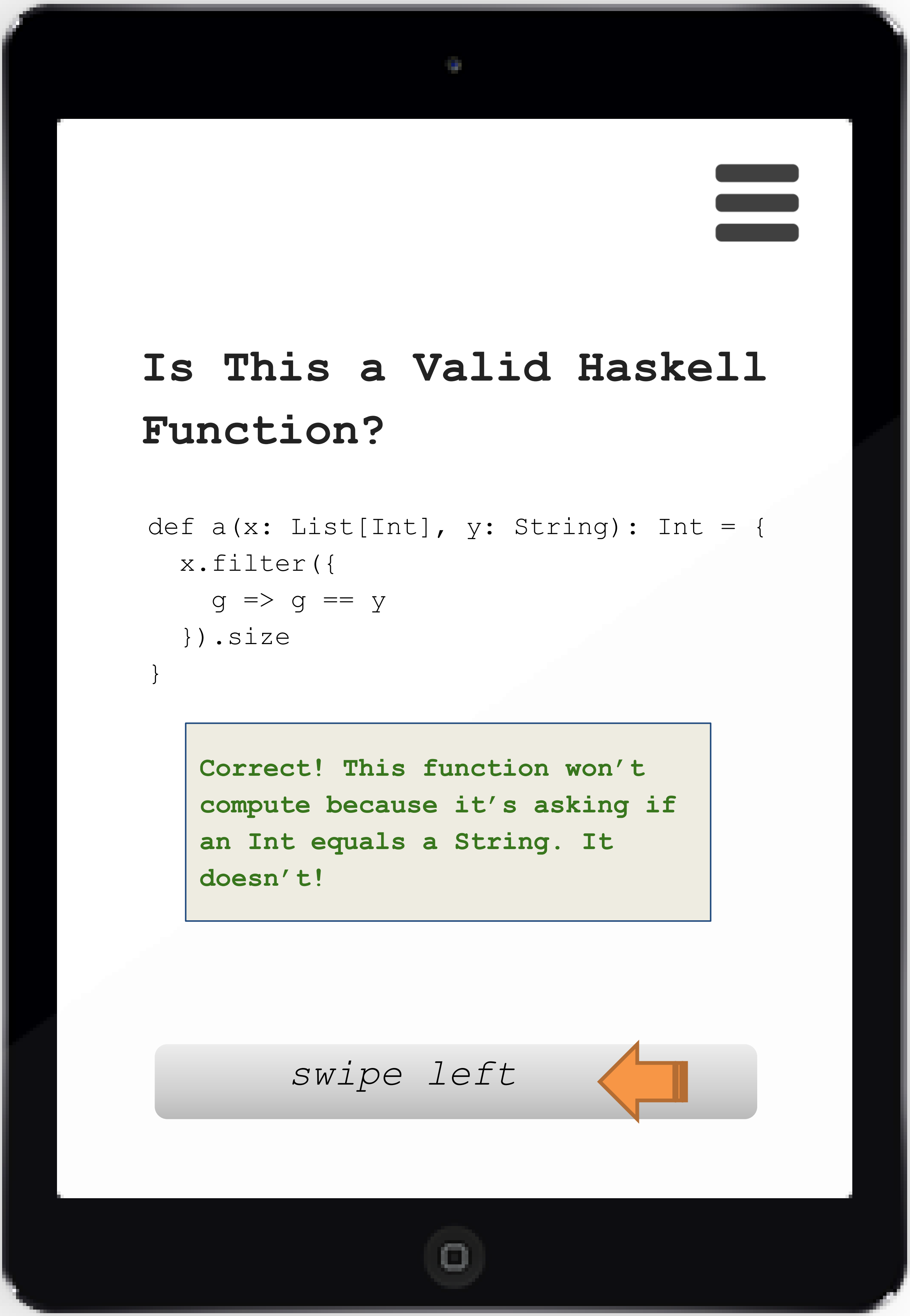
```
def b(x: List[Int], y: String): Int = {
  x.map({
    g => y + g.toString
  }).size
}
```

**Correct! They're using toString to convert an integer to a string before concatenating it.**



# Is This a Valid Haskell Function?

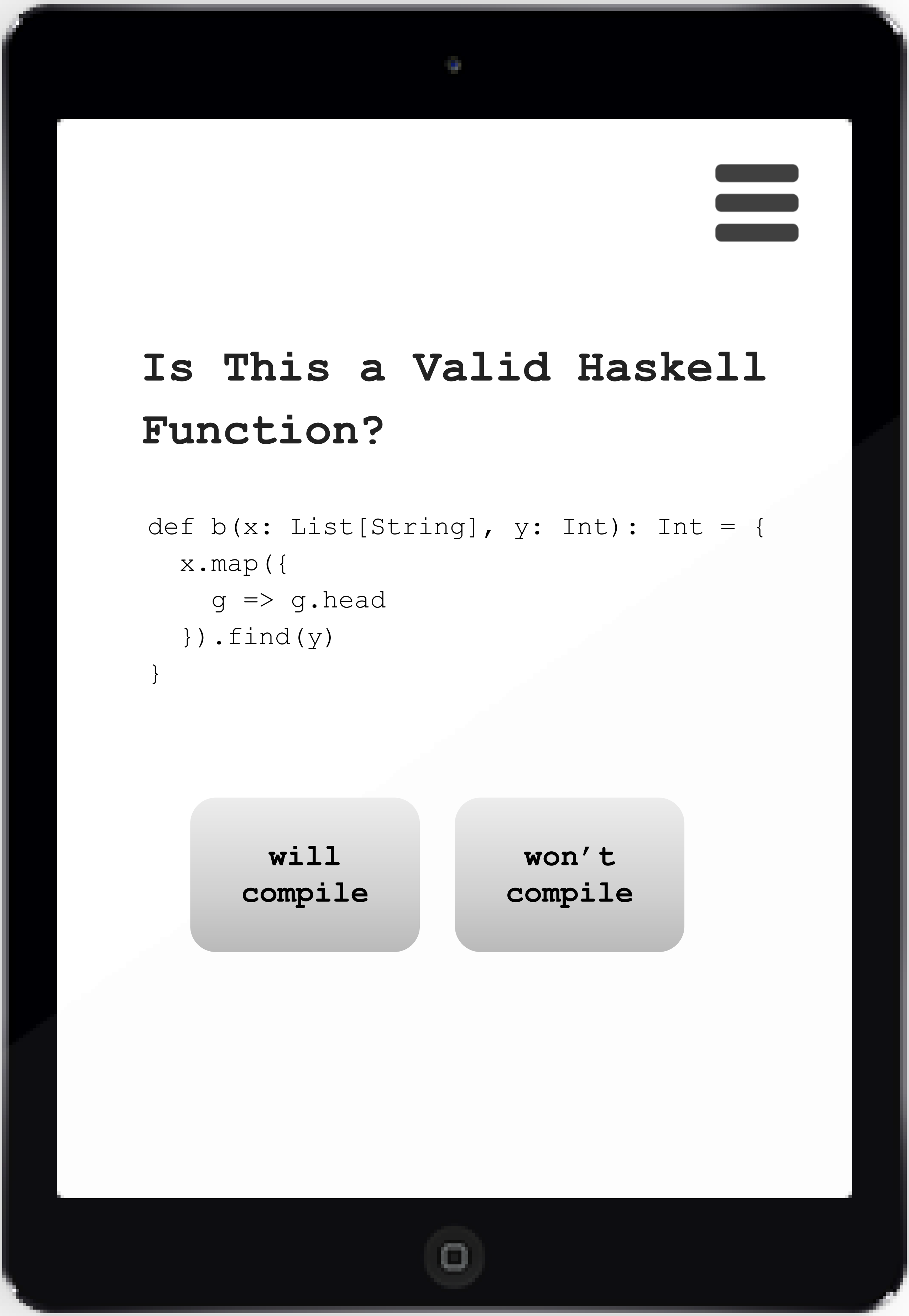
```
def a(x: List[Int], y: String): Int = {
  x.filter({
    g => g == y
  }).size
}
```



# Is This a Valid Haskell Function?

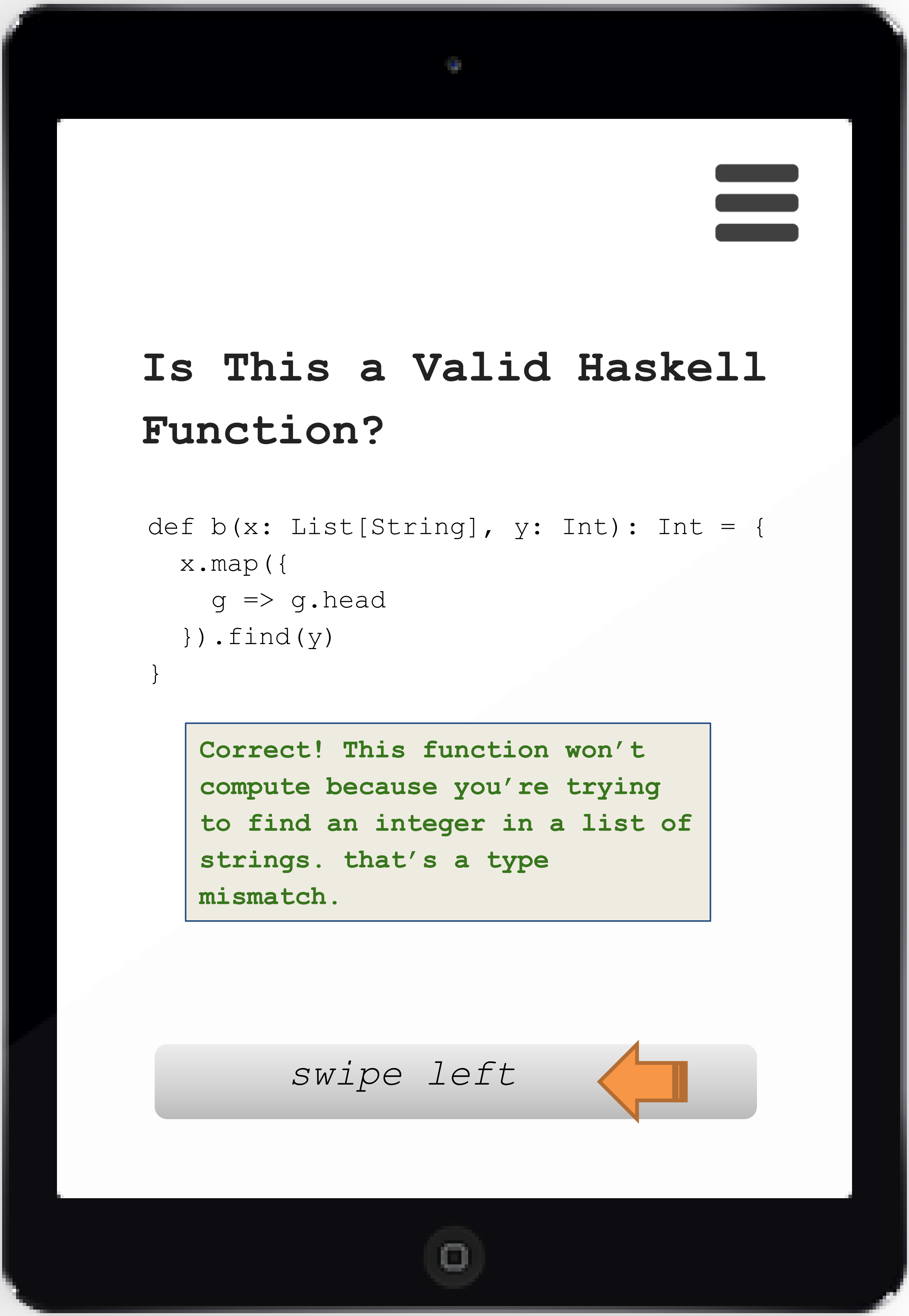
```
def a(x: List[Int], y: String): Int = {
  x.filter({
    g => g == y
  }).size
}
```

Correct! This function won't compute because it's asking if an Int equals a String. It doesn't!



## Is This a Valid Haskell Function?

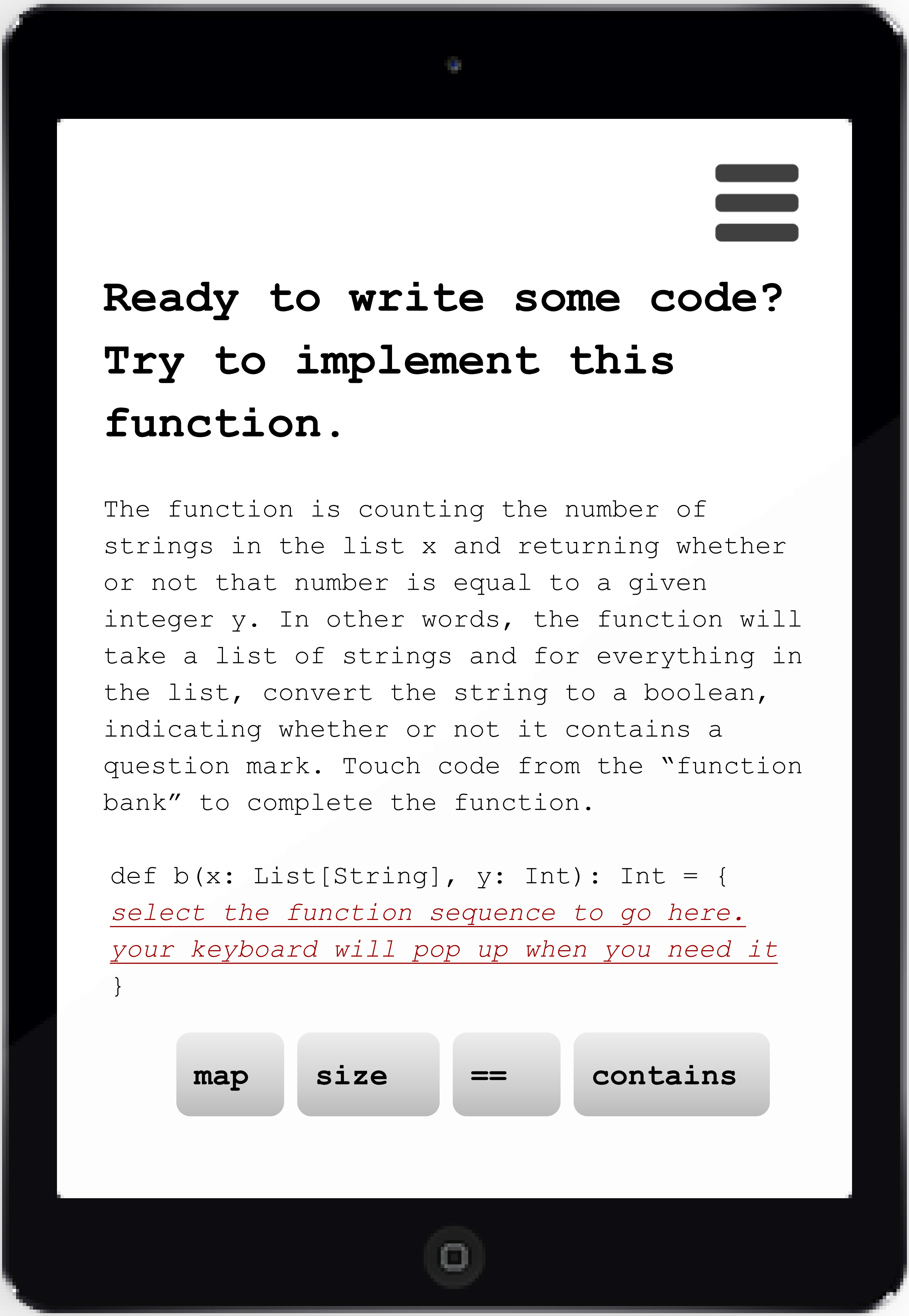
```
def b(x: List[String], y: Int): Int = {
  x.map({
    g => g.head
  }).find(y)
}
```



# Is This a Valid Haskell Function?

```
def b(x: List[String], y: Int): Int = {
  x.map({
    g => g.head
  }).find(y)
}
```

Correct! This function won't compute because you're trying to find an integer in a list of strings. that's a type mismatch.



# Ready to write some code? Try to implement this function.

The function is counting the number of strings in the list x and returning whether or not that number is equal to a given integer y. In other words, the function will take a list of strings and for everything in the list, convert the string to a boolean, indicating whether or not it contains a question mark. Touch code from the “function bank” to complete the function.

```
def b(x: List[String], y: Int): Int = {
select the function sequence to go here.
your keyboard will pop up when you need it
}
```