Solution Review: Max with Nested Functions

In the following lesson, we will go over the solution of the challenge: Max with Nested Functions.

We'll cover the following

Task

In this challenge, you had to create a nested function `max` which would help its parent function `mainMax` to compute the maximum of three numbers.

Solution

A skeleton of the `mainMax` function was already provided for you. Let's look it over.

```
def mainMax(a: Int, b: Int, c: Int): Int = {

}
```

`mainMax` takes three parameters of type `int` and returns a value of type `Int`.

Let's go over the step-by-step process for writing the `max` function.

- `max` is intended to break down the bigger problem into a smaller one. While `mainMax` returns the maximum of three numbers, `max` returns the maximum of two of them. This means that it will take two parameters of type `Int` and return the greater of the two. To find the maximum of two numbers, a simple `if-else` expression can be used.

```
def max(x: Int, y: Int) = {
  if(x > y) x
    else y
}
```

- As for the return value of `mainMax`, we simply needed to call the `max` function.

The first argument will be one of the three numbers passed to `mainMax` and

the second argument will be the maximum of the remaining two. To get the second argument, we will use the `max` function again as it returns the maximum of two numbers.

```
max(a,max(b,c))
```

You can find the complete solution below:

> You were required to write the code from **line 1** to **line** 7.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
def mainMax(a: Int, b: Int, c: Int): Int = {
  def max(x: Int, y: Int) = {
    if(x > y) x
    else y
  }
  max(a,max(b,c))
}

// Driver code
print(mainMax(1,5,9))
```

In the next lesson, we will learn about lexical scope.