Solution Review: Sum of Lists

In the following lesson, we will go over the solution of the challenge: Sum of Lists.

We'll cover the following

Task

In this challenge, you had to create a recursive function `sum` which sums together all the integers in a List.

Solution

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

```
def sum(numberList: List[Int]): Int = numberList match {

}
```

`sum` takes a single parameter of type `List[Int]` and returns a value of type `Int`.

The function body of `sum` consists of a `match` expression. You needed to write the cases required for `sum` to execute correctly. The first case would be our base case which is if the List is empty. In this case, `sum` would return **0**.

```
case Nil => 0
```

The second case is if the List is not empty. In this case, we separate the first element of the List from the rest of the elements. Remember that the `tail` method returns the complete List excluding the first element. We can use `tail` to get the first element.

```
case x :: tail
```

In the above code, the first element is stored in `x`.

If the case is that the List is not empty, we need to sum the first element with the recursive call of `sum` which is passed the `tail` of the List.

```
case x :: tail => sum(tail)
```

You can find the complete solution below:

> You were required to write the code from **line 1** to **line 4**.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
def sum(numberList: List[Int]): Int = numberList match {
  case Nil => 0
  case x :: tail => x + sum(tail)
}

// Driver code
val list = List(1,2,3,4,5)
print(sum(list))
```

Alternate Solution

The alternate solution to this challenge was using the `if-else` expression. The logic behind the code is pretty much the same as the `match` solution. The only difference is that the `if-else` solution takes a second parameter which is the **length - 1** of `list`. In other words, if our list has **5** elements, we need to pass **4** to `sum`.

The first case is represented by `if` which is now checking if we have reached the start of the list (`index < 0`). The second case is represented by `else` which is adding the last element of the list (element at index **4**) to the `sum` call `sum(numberList, index-1)`. `sum` will now look at the list starting from the second to last index, i.e., **3**. This will continue until we have reached the first element of the

list.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
//Alternative solution
def sum(numberList: List[Int], index: Int): Int = {
  if(index < 0 ) 0
  else numberList(index) + sum(numberList, index-1)
}

// Driver code
val list = List(1,2,3,4,5)
print(sum(list,4))
```

In the next lesson, we will be looking at an example which will be carried forward for the next several lessons.