Exercise: Layout

List folding operations

In the provided Elm project, you will find a module `Layout`, where you need to implement three versions of function `layout` with the following signature:

```
layout: List Int -> String .
```

This function receives a list of integer numbers and returns a string layout of its numbers surrounded and separated by slash symbols. That is, `layout [1, 3, 5]` is `"/1/3/5/"` .

You must implement three versions of this function:

- `layoutA` , where you do not use any of the predefined `List` functions but only use recursion.
- `layoutB` , defined as `layoutB = List.foldl accLeft initLeft` , where you only need to implement the helpers `accLeft` and `initLeft` .
- `layoutC` , defined as `layoutC = List.foldr accRight initRight` , where you only need to implement the helpers `accRight` and `initRight` .

Note that in the last two cases you may not change the already given definitions of `layoutB` and `layoutC` . You must only implement the helpers.

The only predefined Elm functions you are allowed to use in the implementation of `layoutA` and the helpers of `layoutB` and `layoutC` are the string concatenation operator `++` and the string conversion function `String.fromInt` .

The provided Elm project contains also a module `Tests` with unit tests corresponding to all three versions of the function.

End of exercise