Extra foldr/foldl practice: **Solutions**

You should be able to solve problems similar to a) – f) on this handout for exams in this course. Questions g) – j) are beyond what we'd expect you to work out by hand for an exam, but they're great practice (and show some really crazy behavior!)

a. `foldr (-) 0 [8,7,6,5]`

```
(8 – (7 – (6 – (5 – 0)))) = 2
```

b. `foldl (-) 0 [8,7,6,5]`

```
((((0 – 8) – 7) – 6) – 5) = -26
```

c. `foldr (:) [] [1,2,3,4,5]`

```
1 : (2 : (3 : (4 : (5 : [])))) = [1,2,3,4,5]
```

d. `foldl (:) [] [1,2,3,4,5]`

```
((((([] : 1) : 2) : 3) : 4) : 5) = error ([] : 1 doesn't make sense)
```

e. `foldr (/) 0 [1, 2, 3, 4, 5]`

```
1 / (2 / (3 / (4 / (5 / 0)))) = Infinity
```

f. `foldl (/) 0 [1, 2, 3, 4, 5]`

```
(((((0 / 1) / 2) / 3) / 4) / 5) = 0
```

The following involve function composition; remember f . g x is the same as f(g(x)), where f and g are functions:

g. `foldr ((++) . map (* 2)) [] [[1,2,3],[4,5,6],[7,8,9]]`

(This one does what you might expect)

```
(map (* 2) [1,2,3]) ++ ((map (* 2) [4, 5, 6]) ++ ((map (* 2) [7,8,9])
++ []))) = [2,4,6,8,10,12,14,16,18]
```

h. `foldl ((++) . map (* 2)) [] [[1,2,3],[4,5,6],[7,8,9]]`

(This one does something really bizarre)

```
(map (* 2) ((map (* 2) ((map (* 2) []) ++ [1,2,3])) ++ [4,5,6])) ++
[7,8,9] = [4,8,12,8,10,12,7,8,9]
```

The syntax for this is probably pretty confusing. If you draw a tree to represent the computation, the root node of every subtree is `(++) . (map (* 2))` – you can think of the `(map (* 2))` as being applied to the result of the left subtree's computation, and then the `(++)` appends the right list to that result. This works the same way for a foldr tree, except the left subtree will just be a single list element.

```
i. foldr ((++) . reverse) [] ["Spyro", "the", "Dragon"]
(reverse “Spyro”) ++ ((reverse “the”) ++ ((reverse “Dragon”) ++ []))
= “orypSehtnogarD”
```

```
j. foldl ((++) . reverse) [] ["Spyro", "the", "Dragon"]
(reverse (reverse ((reverse []) ++ “Spyro”)) ++ “the”)) ++ “Dragon” =
“ehtSpyroDragon”
```