# More Exercises: Functions

## Add with Zip

Implement a Haskell function which accepts 2 arrays, uses **zip** (**NOT** **zipWith**) and adds each element of the first array with each element of the second.

This is quite easy with **zipWith** (**zipWith (\x y -> x + y) firstList secondList**)… For the purpose of exercising, try to do it with **zip** and **WITHOUT** implementing any custom functions.

### Examples

|  |  |
| --- | --- |
| **Arrays** | **Function result** |
| [25, 43, 11, 28, 44]  [15, -3, 29, 12, -4] | [40, 40, 40, 40, 40] |

## Broken Zipper

Implement a Haskell function which accepts 2 arrays and adds each element of the first array with each element of the second.

There is one additional condition though, if **one** of the **arrays’s elements** is a **negative number**, that means that’s a broken zipper element, and a **zero** should be left there instead. (like a broken zipper, leaving a blank disconnected space).

For the purpose of exercising, try to do it **WITHOUT** implementing any custom functions.

### Examples

|  |  |  |
| --- | --- | --- |
| **Arrays** | **Function result** | **Comment** |
| [23, -6, 48, 54, 12, -5]  [15, -3, 55, 3, -4, 6] | [38,0,103,57,0,0] | There are 3 cases when one or more elements are negative, a zero is presented in that case. |