# SWA1 Exercises 4

## Exercise 4.1 – Map, filter, reduce

Rewrite the following functions using map, filter and reduce.

```
function names(persons) {
  let ns = []
  for(let i = 0; i < persons.length; i++) {
    ns.push(person[i].name)
  }
  return ns
}

function adults(persons) {
  let as = []
  for(let i = 0; i < persons.length; i++) {
    if (persons[i].age >= 18) {
      as.push(persons[i])
    }
  }
  return as
}

function oldest_person(persons) {
  let oldest = null
  for(let i = 0; i < persons.length; i++) {
    if (!oldest || persons[i].age > oldest.age) {
      oldest = person[i]
    }
  }
  return oldest
}

function total_salaries_of_seniors(employees) {
  let total = 0
  for(let i = 0; i < persons.length; i++) {
    if (persons[i].age >= 60) {
      total += persons[i].salary
    }
  }
  return total
}
```

## Exercise 4.2 – Immutability

Write immutable versions of the factory functions from exercise 1.2.

## Exercise 4.3 – Closures

a) Create a function that takes a value, n, and returns a function that raises its argument to the power of n.
b) Create a function that returns a function that gives subsequent elements of the Fibonacci sequence.

## Exercise 4.4 – Functional style

Rewrite the classes from exercise 7 as either

a) Immutable classes
b) Functions + data

## Exercise 4.5

Write curried versions of the following functions

```
function add(n, m) {
  return n + m
}

function greater(n, m) {
  return n > m
}

function get(attr, o) {
  return o[attr]
}

function pipe(f, g) {
  return function(x) {
    let r = f(x)
    return g(r)
  }
}
```

Rewrite your solution to `names`(), `adults`() and `total_salariesof_seniors`() using these functions.

## Exercise 4.6

a) Implement a `reduce(array, operator, defaultValue)` function without using the built-in reduce.

b) Implement map and filter using your reduce function from (a). (Hint : `defaultValue = []`)