Lab Exercise 7: Functional Programming: Basic Exercises

Even these supposedly basic exercises are a bit tricky if you have never seen functional programming.

1. Make a function called composedValue that takes two functions f1 and f2 and a value and returns f1(f2(value)), i.e., the first function called on the result of the second function called on the value.

```
function square(x) { return(x*x); }
function double(x) { return(x*2); }
composedValue(square, double, 5); --> 100 // I.e., square(double(5))
```

2. Make a function called compose that takes two functions f1 and f2 and returns a new function that, when called on a value, will return f1(f2(value)). Assume that f1 and f2 each take exactly one argument.

```
var f1 = compose(square, double);
f1(5); --> 100
f1(10); --> 400
var f2 = compose(double, square);
f2(5); --> 50
f2(10); --> 200
```

3. Make a function called “find” that takes an array and a test function, and returns the first element of the array that “passes” (returns non-false for) the test. Don’t use map, filter, or reduce.

```
function isEven(num) { return(num%2 == 0); }
isEven(3) --> false
isEven(4) --> true
find([1, 3, 5, 4, 2], isEven); --> 4
```

4. Recent JavaScript versions added the “map” method of arrays, as we saw in the notes and used in the previous set of exercises. But, in earlier JavaScript versions, you had to write it yourself. Make a function called “map” that takes an array and a function, and returns a new array that is the result of calling the function on each element of the input array. Don’t use map, filter, or reduce.

```
map([1, 2, 3, 4, 5], square); --> [1, 4, 9, 16, 25]
map([1, 4, 9, 16, 25], Math.sqrt); --> [1, 2, 3, 4, 5]
```

Hint: remember the push method of arrays.