January 06

Challenges

1. ***Implement, as best as you can, the identity function in your favorite language (or the second favorite, if your favorite language happens to be Haskell).***

   I have chosen JavaScript as my language of implementation, even though it is not my prefered language. However JavaScript conveniently supports the functional programming paradigm and hence implementing anonymous functions is easy, and the language allows higher order functions.

```
1
2   const identity = (e) => e;
3
```

Listing 1: Identity function implemented in JavaScript

2. ***Implement the composition function in your favorite language. It takes two functions as arguments and returns a function that is their composition.***

```
1
2   const composition = (f, g) =>{
3       (...args) => f(g(...args))
4   };
5
```

Listing 2: Composition function

```
1
2   const composition = (f, g) => {
3       return (...args) => {
4           let x = g(...args);
5           let y = f(x);
6
7           /* Loging of computation */
8           console.log("g(%s) => %s", args, x);
9           console.log("f(%s) => %s", x, y);
10          console.log("f(g(%s)) => %s", args, y);
11
12          return y;
13      }
14  };
15
```

Listing 3: Composition function with logging

3. ***Write a program that tries to test that your composition function respects identity.***

   It is a bit difficult to deduce what is meant by this question, so I have to let my assumptions be clear. I understand this question in such a way so that I am to write a program that checks if function when composed with the identity function, is the same before and after function composition. I have written such a function down below. However, in JavaScript it will never hold that an arbitrary function composed with the identity function is the same. It is not intentionally equal. The composition function returns a new function that is located in a new location in *memory*. Extensionally however, it is the same function! If we were to write a program that would check for this kind of equality, it would have to check every single input and compare the result. Resulting in a non-terminating program.

```
1    const assertSameFunction = (f) => {
2        return f === composition(f, identity);
3    };
4
```

Listing 4: Function for asserting intentional equality between functions

4. ***Is the world-wide web a category in any sense? Are links morphisms?***

   Yes, given that a link is a morphism/function/arrow. To be a node in the world-wide web, you have to be connected to it. Hence there is a set of morphism from A to B, for every pair of computers A and B. If you have a link from computer A to computer B, and another link from computer B to computer C, then there must be a - the composition or combination of the two links - that goes from A to C. In addition every computer is connected to itself using a feedback loop, that we in this case can call the identity morphism.

5. ***Is Facebook a category, with people as objects and friendships as morphisms?***

   No, because for every pair of people A and B, there is not a guaranteed set of morphisms(or arrows) from A to B. Meaning that if you have two arbitrary people A and B, there is a possibility that there is not path from A to B using the friendship as edges. The easiest example a person that has no friends. No other person will have a set of morphism (composition) to this person. In addition to that, no person is friends with themselves.

6. ***When is a directed graph a category?***

   A directed graph is a category when for every pair of nodes A and B, there is a set of morphism from A to B. For every object A, there is an identity morphism that maps A to itself.