COMP 333
Spring 2024

Higher-Order Functions in JavaScript

1.) Write a function named `callMe` which takes a function `foo` and will call `foo`.

```
function callMe(foo) {
  foo();
}
```

2.) Write a function named `indirectIf` which takes a boolean and two functions. If the boolean is `true`, call the first function. Otherwise, call the second function.

```
function indirectIf(booleanValue, f1, f2) {
  if (booleanValue) {
    f1();
  } else {
    f2();
  }
}
```

3.) Write a function named `indirectWhile` which takes two functions. The first function returns a boolean, and the second function returns nothing. `indirectWhile` should call the first function, and if the result is `true`, it should call the second function followed by a recursive call to itself with the same parameters. If the first function returns `false`, `indirectWhile` does nothing.

```
function indirectWhile(returnsBool, returnsNothing) {
  if (returnsBool()) {
    returnsNothing();
    indirectWhile(returnsBool, returnsNothing);
  }
}
```

4.) Write a function named `wrapAdd` which takes a function (which itself takes one parameter) and an integer. `wrapAdd` should return a new function which takes a parameter, and will add this parameter to the integer before calling the passed function. For example:

```
function returnParam(param) { return param; }

let f = wrapAdd(returnParam, 5);
let x = f(2); // x = 7
let y = f(3); // y = 8

function wrapAdd(f, integer) {
  return function(a) {
    return f(a + integer);
  }
}
```