Class Activity #2

Q1:

```
function fibonacci(n){
  if(n===0) return 0;
  if(n===1) return 1;
  return fibonacci(n-1)+fibonacci(n-2);
}
```

Q2: *There are many ways to solve this problem, notice, however, that the solution without array consumes less memory.*

```
function fibonacci(n) {
  var fib = [0, 1];
  for (var i = 2; i <= n; i++) {
    fib[i] = fib[i-1] + fib[i-2];
  }
  return fib[n];
}
function fibonacci(n) {
  if (n === 0) return 0;
  if (n === 1) return 1;
  let total2 = 0;
  let total1 = 1;
  for (let i = 2; i <= n; i++) {
    let temp = total2;
    total2 = total1;
    total1 = total1 + temp;
  }
  return total1;
}
```

Q3: *On the first print: the step-by-step evaluation would be from left to right (the underline parts, are the operands that get returned after each evaluation):*

```
"" || "" || "JCSS" || "Anonymous"

      "" (false) || "" (false) || "JCSS" || "Anonymous"

            "" (false) || "JCSS" (true) || "Anonymous"
```

The execution stops right here because **||** *is short-circuited (it finds the first "truthy" value) – in this example, "JCSS".*

On the second example, the step-by-step evaluation is also left-to-right, but the difference here is that && takes precedence. That's the execution flow:

```
"" || "" || "JCSS" (true) && "Anonymous" (true)

      ""  (false) || ""  (false) || "Anonymous"

          ""  (false)  || "Anonymous"(true)
```

→ *from this step and on, execution is back left-to-right*

The execution finishes with "Anonymous".