National University of Singapore
School of Computing
CS1101S: Programming Methodology
Semester I, 2016/2017

Recitation 2
Recursion & Iteration

Iterative vs Recursive Processes

A function that calls itself gives rise to an iterative process if the recursive call is always the last operation that the function carries out. This means that the result of the recursive call is also the result of the function.

```
function f(x, y) {
    if (x === 0) {
       return y;
    } else {
       return f(x - 1, y + 1);
    }
}
```

The body of function `f` has one recursive call. The result of the recursive call is the result of the function. This means that the function gives rise to an iterative process.

```
function g(x) {
    if (x === 0) {
       return 0;
    } else {
       return 1 + g(x - 1);
    }
}
```

The body of function `g` has one recursive call. The result of the recursive call is not the result of the function. When the recursive call returns, its result needs to be added to 1. This operation is called a "deferred operation". The presence of such a deferred operation means that the corresponding process is a recursive process.

We say the function gives rise to a recursive process.

The Fibonacci Series

Leonardo Pisano Fibonacci (12th century CE), was interested in the sequence:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$$

Each number is the sum of the previous two.

Definition The function *fib* maps 0 to 0, 1 to 1, and every subsequent natural number n to the sum of the two previous Fibonacci numbers: $fib(n-2) + fib(n-1)$.

Problems:

1.
```
function fact(n) {
    if(n === 0) {
        return 1;
    } else {
        return n * fact(n-1);
    }
}
```

   Is this function giving rise to a recursive or iterative process?

2. How many steps does it take to compute the result of this function when applied to the number 5?

3. How much space is required to compute the result of this function when applied to the number 5? Count the number of deferred operations.

4. Write an iterative version of `fact`.

5. How many steps does it take to compute the result of your iterative function when applied to the number 5?

6. How much space is required to compute the result of your iterative function when applied to the number 5? Count the number of deferred operations.

7. Consider the following implementation of the Fibonacci function:

```
function fib(n) {
    return n <= 1 ? n
                  : fib(n - 1) + fib(n - 2);
}
```

   Is this function giving rise to a recursive or iterative process?

8. Write an iterative version of `fib`.

9. How many steps does it take to compute the result of your iterative function when applied to the number 5?

10. How much space is required to compute the result of your iterative function when applied to the number 5? Count the number of deferred operations.