National University of Singapore
School of Computing
CS1101S: Programming Methodology
Semester I, 2013/2014

# Recitation 3
# Higher-Order Functions

## From Lecture

Implementation of `pair`, `head` and `tail`:

```
function pair(x,y) {
    return function(m) { return m(x, y); }
}

function head(z) {
    return z(function(p, q) { return p; });
}

function tail(z) {
    return z(function(p, q) { return q; });
}
```

## Definitions

The following are two higher-order functions discussed in lecture:

```
function sum(term, a, next, b) {
    if(a > b) {
        return 0;
    } else {
        return term(a) + sum(term, next(a), next, b);
    }
}

function fold(op, f, n) {
    if(n === 0) {
        return f(0);
    } else {
        return op(f(n), fold(op, f, n - 1));
    }
}
```

Note: it is not necessary to memorize these defintions, or even the names of these functions. Definitions of such functions (if they are used) will be given in an Appendix for examinations. What is most important is that students must be able to read the definition for such a function and understand what it does.

# Problems:

1. Evaluate the return values of the following sets of statements:

   (a) var x = 12; x;
   **Answer:** 12

   (b) var x = 12; (function() x = 15; )(); x;
   **Answer:** 15

   (c) var x = 20; (function() var x = 15; )(); x;
   **Answer:** 20

2. Write a function `my_sum` that computes the following sum, for $n \geq 1$:

$$1 \times 2 + 2 \times 3 + \cdots + n \times (n+1)$$

   **Answer:**

```
function my_sum(n) {
    if(n === 1) {
        return 2;
    } else {
        return (n * (n + 1)) + (my_sum(n - 1));
    }
}
```

3. Is the function `my_sum` as defined in Question 1 above a recursive process or an iterative process? What is the order of growth in time and in space?

   **Answer:** Recursive. Time: $O(n)$, space: $O(n)$.

4. If your answer in Question 2 is a recursive process, re-write `my_sum` as an iterative process. If your answer in Question 2 is an iterative process, re-write `my_sum` as a recursive process.

   **Answer:**

```
function my_sum(n) {
    function my_sum_iter(counter, sum) {
        if(counter === 0) {
            return sum;
        } else {
            return my_sum_iter(counter - 1, sum + (counter * (counter + 1)));
        }
    }
    return my_sum_iter(n, 0);
}
```

5. We can also define `my_sum` in terms of the higher-order function `sum`. Complete the definition of `my_sum` below. You cannot change the definition of `sum`; you may only call it with appropriate arguments.

```
function my_sum(n) { return sum(<T1>, <T2>, <T3>, <T4>); }
```

   **Answer:**

```
T1: function(n) { return n* (n + 1); }


T2: 1


T3: function(n) { return n + 1; }


T4: n
```

6. Suppose instead we define `my_sum` in terms of the higher-order function `fold`. Complete the definition of `my_sum` below.

   ```
   function my_sum(n) { return fold(<T1>, <T2>, <T3>); }
   ```

   **Answer:**

   ```
   T1: function(a, b) { return a + b; }


   T2: function(n) {(if (= n 0)
           if(n === 0) { return 0; }
           else{ return n * (n + 1); }
       }
   T3: n
   ```

7. Write an iterative version of `sum`.

   **Answer:**

   ```
   function sum(term, a, next, b) {
       function sum_iter(a, total) {
           if(a > b) {
               return total;
           } else {
               return sum_iter(next(a), total + term(a));
           }
       }
       return sum_iter(a, 0);
   }
   ```

8. **Homework:** Write an iterative version of `fold`.