Recitation Session, Sept 27 2017

Please do not write on this sheet of paper
And do not use laptops during the session

We will work on tail recursion in this session.

Exercise 1: Factorial

Recall the factorial function that you saw in class

```
def factorial(n: Int): Int = if (n <= 0) 1 else n * factorial(n - 1)
```

Define a tail recursive version of it

```
def factorial(n: Int): Int = fact(n, 1)
@tailrec
def fact(n: Int, acc: Int): Int = ???
```

What would be the advantage of making `fact` an inner function to factorial?

Exercise 2: Sum of elements on a list

Define a function that takes a list of integers and sums them. You can use the functions `head`, `tail`, and `isEmpty` on lists, as you have seen for your homework.

```
def sumList(ls: List[Int]): Int = ???
```

Convert your definition into a tail-recursive one.

Exercise 3: Fast exponentiation

Fast exponentiation is a technique to optimize the exponentiation of numbers:

$b^{2n} = (b^2)^n = (b^n)^2$
$b^{2n+1} = b * b^{2n}$

Define a function that implements this fast exponentiation. Can you define a tail recursive version as well?

```
def fastExp(base: Int, exp: Int): Int = ???
```

Exercise 4: Tail recursive Fibonacci

Define a function that computes the nth Fibonacci number. Can you define a tail recursive version as well? The Fibonacci recurrence is given as follows:

fib(n) = 1 | n = 0, 1
fib(n) = fib(n - 1) + fib(n - 2) | otherwise

```
def fibonacci(n: Int): Int = ???
```