# Recitation 1 – *Asymptotics and Divide and Conquer (Worksheet)*

Parallel and Sequential Data Structures and Algorithms, 15-210 (Spring 2015)

*January 13th, 2015*

## 1 Welcome to 15-210!

- Lab 0 – *SignalLab* will be released this Friday, January 16th.
- We will be using SML almost exclusively in this course. Please take some time this week to solidify your knowledge of the language.
- You should refer to the class website for course policy, schedule, library documentation... well, everything, really.

  http://www.cs.cmu.edu/~15210/
- Make sure you have no problems logging in to Piazza and Autolab!

## 2 Asymptotics

**Definition of Big-$O$:** Consider functions $f : \mathbb{N} \to \mathbb{R}_+$ and $g : \mathbb{N} \to \mathbb{R}_+$. We say that $f \in O(g)$ if and only if there exist constants $N_0 \in \mathbb{N}$ and $c \in \mathbb{R}_+$ such that for all $n \geq N_0$, $f(n) \leq c \cdot g(n)$.

- Use the definition of Big-$O$ to prove that $n^2 \in O(n^3 - 100)$.

- Order the following functions by their Big-$O$ class from smallest to largest.
  1. $(n!)^2$
  2. $100n + \log n$
  3. $3n^2$
  4. $(n^2)!$
  5. $1/n$
  6. $\log_{\left(100^{100^{100^{100}}}\right)} n$

## 3 Divide and Conquer

A string $S$ is **closed** if it contains only '(' and ')', and is one of the following:

- The empty string.
- The concatenation of two closed strings: $S_1S_2$
- A single closed string $S_0$ surrounded by a pair of *matched* parentheses: $(S_0)$

Instead of strings, we'll be manipulating things of type `paren seq`, where

```
datatype paren = OPAREN (* open *) | CPAREN (* close *)
```

We'll also be using `showt : 'a seq → 'a treeview`, which splits a sequence approximately in half.

```
datatype 'a treeview =
    EMPTY                       (* empty *)
  | ELT of 'a                   (* single element *)
  | NODE of ('a seq * 'a seq)   (* split in half *)
```

And finally, we have a function `par` which runs two functions in parallel:

```
val par : (unit → 'a) * (unit → 'b) → ('a * 'b)
```

**Your task:** Complete the implementation of the predicate `closed` below, which checks to see if a `paren seq` is closed under the definition given above.

*Note:* This might be challenging. If you fail, don't worry. We'll be covering divide and conquer techniques extensively in the coming weeks.

*Hint:* Consider representing a subsequence of parentheses with two numbers $(i, j)$, where the subsequence begins with $i$ unmatched close parens and ends with $j$ unmatched open parens.

```
fun closed (S : paren seq) : bool =
  let
    fun closed' S =
      case showt S of
        EMPTY       ⇒ ____________________________

      | ELT OPAREN   ⇒ ____________________________

      | ELT CPAREN   ⇒ ____________________________

      | NODE (L, R) ⇒
        let

          val (____________________________, ____________________________) =
            par (fn () ⇒ closed' L, fn () ⇒ closed' R)
        in


          ____________________________________________________
        end
  in
    closed' S = ____________________________
  end
```