CpSc 418 **Mini-Assignment 3** Due: Feb. 8, 2016, 10:00am

20 points. Please submit your solution using the handin program. Submit your solution as
cs418 mini3

Your submission should consist of the one file called `mini3.erl` or `mini3.txt` sepending on which problem you choose to solve.

Submit a solution to any **one** of the three questions below.

1. **Erlang Review** Let `LL` be a list-of-lists.

   (a) **(5 points)** Write an Erlang function, `is_lol(X)` that returns true iff `X` is a list and every element of `X` is a list. Note that `is_lol([]) -> true` because `[]` is a list and every element of `[]` (there aren't any) is a list as well.

   (b) **(5 points)** Write an Erlang function, `is_rect(X)` that returns true iff `X` is a list and all elements of `X` are lists of the same length.

   (c) **(10 points)** Write an Erlang function, `transpose(X)` such that

   - `transpose([]) -> []`;
   - If `X` is a list where every element of `X` is the empty list, then `transposeX -> []`;
   - If `is_rect(X)`, `X` is non-empty, and the elements of `X` are non-empty lists, let `N1 = length(X)`, and `N2 = length(hd(X))`. Then, `transposeX -> Y` where `is_rect(Y)`, `length(Y) =:= N2`, every element of `Y` is a list of length `N1`, and `lists:nth(I, lists:nth(J, Y)) =:= lists:nth(J, lists:nth(I, X))` for all `0 < I =< N1` and `0 < J =< N2`.
   - If `not is_rect(X)` then `transpose(X)` fails with an error.

   Note: I will not ask you to write this much code on the midterm. OTOH, I might state a similar problem, write a partial solution, and ask you to fill-in 3-5 lines.

2. **Reduce and/or Scan** Let

```
goodness([]) -> 0;
goodness([good | Tl]) -> 1 + goodness(Tl);
goodness([bad | Tl]) -> -1 + goodness(Tl);
goodness([_ | Tl]) -> goodness(Tl);
```

   (a) **(10 points)** Write a function, `stop_while_your_ahead(List)` that returns `{I, G}` where `0 =< I =< length(List)` maximizes `G = goodness(element(1, lists:split(I, List)))`. If there's a tie, return the smallest such `I`.

   (b) **10 points** Use `wtree:reduce` to write
   `stop_while_your_ahead_par(W, Key)` that returns `{I, G}` as above for the list associated with `Key` that is distributed across the workers of codeW.

3. **Speed-Up** Consider a problem that can be executed sequentially in time $(20ns) * N \log_2 N$, where N is the size of the input problem. A parallel implementation with P processors takes time:

$$(20\text{ns}) * \frac{N}{P} \log_2 \left(\frac{N}{P}\right) + \left(10\text{ns} * \frac{N}{P} + \lambda\right) \log_2(P)(\log_2(P) + 1)/2$$

   Where λ is the cost of a communication action. For this problem, assume $\lambda = 5\mu$s.

   (a) **(10 points)** Plot speed-up as a function of `P` for `P` chosen from 4, 16, 64, and 256, and for `N` chosen from 2^{10}, 2^{20}, and 2^{30}.

   (b) **(10 points)** How large must N be to get a speed-up of at least $P/2$ for each of the values of P suggested above? You only need to consider power-of-two values for N, but you do need to consider powers of two other than the ones listed for part a.

Why?

Review for the midterm.