CpSc 418 **Mini-Assignment 1** Due: Jan. 8, 2016, 10:00am

30 points.

- This mini-assignment is optional – it will only contribute to your mini-assignment percentage (both the numerator and denominator) if you submit a solution. It is intended to be easy; so, it will probably raise your average a little.
- If you are on the wait list for CpSc 418, please note that I will give priority to students who submit solutions to mini assignments 1 and 2. Given the number of students on the wait list and the number of slots that are likely to open up, this means that those who do these mini-assignments will have a much greater chance of getting into the class than those who don't.

Please submit your solution using the handin program. Submit your solution as
cs418 mini1
Your submission should consist of the one file called `mini1.erl`.

You can get a template for `mini1.erl` and a test module, `mini1_test.erl` here. The test cases in `mini1_test.erl` are not exhaustive. If your code doesn't work with these, it will almost certainly have problems with the test cases used for grading. The actual grading will include other tests as well.

The Questions

1. `who_am_i/0`, 3 points.
   Write an Erlang function with no arguments that returns a tuple of three elements. Let `{Name, StudentNumber, PreferredEmail}` be this tuple, where:
   (a) `Name` is an Erlang string – your name.
   (b) `StudentNumber` is an Erlang integer – your student number.
   (c) `PreferredEmail` is an Erlang string – the e-mail address to which you want your grades and other class information sent.
2. `registration`, 2 points.
   Write an Erlang function with no arguments that returns the atom
   - `wait` if you are on the wait list for CpSc 418;
   - `enrolled` if you are already registered; or
   - `audit` if you are auditing.
3. **Pattern Matching**, 5 points.
   Consider the following function to flatten a list (it's in the template file for `mini.erl`):

```
flatten(X) ->
   if X == []        -> X;
      is_list(X)     -> flatten(hd(X)) ++ flatten(tl(X));
      not is_list(X) -> [X]
   end.
```

   See `lists:flatten/1` which does the same thing.

   Rewrite `flatten` using pattern matching instead of `if`.

4. **Tail Recursion**, 20 points.
Consider the following function to compute the `N`th fibonacci number:

```
% Let fib(N) denote the N^th fibonacci number.
% Here's a head-recursive implementation
fib_hr(0) -> 0;
fib_hr(N) -> element(1, fib_pair_hr(N)).

% fib_pair_hr(N) -> {fib(N), fib(N-1)}
fib_pair_hr(1) -> {1, 0};
fib_pair_hr(N) ->
   {F1, F2} = fib_pair_hr(N-1), % F1 = fib(N-1), F2 = fib(N-2)
   {F1 + F2, F1}.
```

(a) Write `fib_tr(N)` which computes the `N`th fibonacci number using tail recursion.

(b) The template file includes a function `time_fib` that measures the runtime for `fib_hr` and `fib_tr` for lists of various lengths. Run this function, then describe and explain the trends that you observe.

Note: I wrote `time_fib` to be simple. In particular, it only takes one run for each list length. This means there will be a fairly large variation if you run `time_fib` multiple times. The main trends are still clearly visible. In a couple of weeks, we'll look at how to make better timing measurements.

Why?

Question 1: `who_am_i`
So we know how to communicate with you.

Question 2: `registration`
So we can give the most timely feedback possible to anyone on the class waiting list.

Question 3: Pattern Matching
Get some practice with pattern matching.

Question 4: Tail Recursion
Get some experience with the trade-offs between head- and tail-recursion. Learn to convert a function from a head-recursive version to a tail-recursive one.