CpSc 418 **Mini-Assignment 2** Due: Jan. 11, 2016, 10:00am

30 points. Please submit your solution using the handin program. Submit your solution as
cs418 mini2
Your submission should consist of the one file called `mini2.erl`.

I will post a template for `mini2.erl` and a test module, `mini2_test.erl` here, but that might not happen until Thursday morning. Because these problems are based on examples from *LYSE*, you can cut-and-paste the code from there to get templates.

Building a better refrigerator.

Starting from the final version of the refrigerator example at the end of More on Multiprocessing in *LYSE*, add the following features:

1. **Grocery shopping**, (10 points)
   After going to the grocery store, our brave refrigerator owner comes home with a **bag** of groceries, and puts all of them in the refrigerator (whether or not that makes sense – just ask my wife). We'll represent a bag of groceries as a list (of course) of tuples (why not?). In particular, our list has the form:
   `[{`*What*`,` *HowMany*`}, ...]`
   where *What* is an atom and `HowMany` is an integer. For example, the list
   `[{banana, 5}, {egg, 12}, {apatosaurus, 3}]`
   means I bought five bananas, a dozen eggs, and three apatosaurs and put them all in the refrigerator.

   Write `mini2:fridge3a` and `mini2:store3` based on `mini2:fridge2` and `mini2:store2` from *LYSE* so that `mini2:store3(Pid, List)` stores the items from a list as described above in the refrigerator, and `Pid` is the pid of the refrigerator. Of course, `mini2:store3(Pid, Item)` where `Item` is an atom should store one `Item` in the refrigerator. We want our new refrigerator to be upward compatible with the old one.

2. **Making Sandwiches**, (20 points)
   Let's say I want to make a banana and apatosaurus sandwich. Obviously, I'll go to the refrigerator remove one banana, one apatosaurus, and one bread. If they aren't all in the fridge, I don't want to remove any of them – it's too much of a hassle stuffing the apatosaurs back into the refrigerator.

   Write `mini2:fridge3` and `mini2:take3` based on `mini2:fridge3a` and `mini2:take2` so that `mini2:take3(Pid, List)` removes the items in `List` from the refrigerator if they are present, and returns `{ok, List}`. If there is something missing in the refrigerator, return `{not_found, List}`. Of course, `mini2:take3(Pid, Item)` where `Item` is an atom should remove one `Item` from the refrigerator. Again, we want our new refrigerator to be upward compatible with the old one.

   If for some reason you can't solve question 1, you can build your solution to this problem (i.e. implementations of `mini2:fridge3` and `mini2:take3` on the original code from *LYSE*. To get full credit, your solution must provide `mini2:fridge3`, `mini2:store3`, and `mini2:take3` that support both `store` and `take` of lists of items.

3. **Even More Fun**, (0 points)
   We could take this further. We could add an `inventory` function that returns a list of everything in the fridge. We could make a version of `take` that blocks until the requested items are in the fridge – waiting for another process to perform the `store` operation– "Honey, could you run down to IGA and pick up some more apatosaurs?".

   BUT, this is a **mini** assignment; so, we'll stop here.

Why?

Get some experience with Erlang processes, sending and receiving messages, and more pattern matching. Make sure you've read the stuff on processes from *LYSE* so I can build on that in lecture.