MA 151: Homework #4

due Tuesday October 3

Written problems

In each of these, simplify the expressions step-by-step to get the final value. If there is an error, say exactly what the problem is. If the function gives an infinite loop, explain in general terms what the output will be. You should show enough detail to make it clear that you know what is going on. In all cases, you should be able to check your answer by typing the expressions into GHCi.

Assume that the following definitions have been loaded into GHCi:

```
f x
 | x > 5      = "nickelback"
 | x < 5      = "creed"
 | otherwise  = "hootie"
 | x == 5     = "dmb"

g (2:as) = 4
g (3:as) = 8
g (a:as) = a+3
```

Now evaluate:

1. `f [3..8]`
2. `[ f x | x <- [3..8] ]`
3. `g [3..8]`
4. `[ g x | x <- [3..8] ]`
5. `g []`

Programming problems

As always, include a type signature for every function you define.

- Write a function called `secretNumber` which takes a number x and compares it to The Secret Number. If x is too big, the answer is "too big!". If it's too small, say "too small!". If x equals The Secret Number, say "right!". The Secret Number is 73.
- Rewrite the `bro` function from Homework #2 using guards (no ifs).
- Rewrite the `ekzer` function from Homework #1 using pattern matching (no `head` or `tail`.)
- Write a function called `spooner` which takes a pair of strings and exchanges their first letters. Use pattern matching- don't use `head` or `tail`. For example:

  `spooner ("hi", "mom")` is `("mi","hom")`

If one of the strings is empty, it should behave like this:

`spooner ("hello","")` is `("ello","h")`

If both strings are empty, just keep them empty. Make sure your function can handle any input without giving a "nonexhaustive patterns" error.

- Write a function called `quadrant` which takes a pair (x, y) and tells which "quadrant" the point (x, y) is in the xy-plane. The answer should be 1, 2, 3, or 4. If the point is on the axes, give the answer 0.
- Write a function called `stringYear` which takes a pair of `Int` and `String` and returns a `String` so that it looks like this:

  `stringYear (2,"Chris")` is `"Chris is a sophomore"`

  `stringYear (1,"Jen")` is `"Jen is a freshman"`

  (use "junior" and "senior" for 3 and 4, and use "graduate student" for anything else.)
- Write a polymorphic function called `match` which takes a list and gives a `Bool` saying whether or not the first two elements are equal. If there is only one element, the answer should always be `True`, and if there are no elements the answer should be `False`. Don't use `if` or `head` anywhere in your definition.
- Write a function called `aLover` which takes a string and changes all vowels (not 'y') to 'a's. Letters should keep their original cases. (Hint: First write a function which does this to a single character, then use that function inside a list comprehension to make `aLover`. You can be inspired by `bigXer` from class.)
- Write a function called `isPrime` which takes a number and answers True or False depending on if the number is prime.
- Write a function called `isComposite` which takes a number and answers True or False depending on if the number is composite.

  A prime number is any number greater than 1 which has no divisors other than 1 or itself. Negative numbers do not count as prime, and neither does 1.

  A composite number is any positive number which has at least 1 divisor other than 1 and itself. Negative numbers do not count as composite.