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

- Write a function called `digit7` which takes an `Int` and returns a `Bool` saying whether or not 7 is one of the digits. (*Hint:* use `show` to turn the number into a list of characters.) Use `digit7` to create a function of no parameters called `square7` which returns the smallest number whose square contains a 7 as a digit.

- Rewrite the `collapser` function from last week using pattern matching (no if's). Call your new version `patCollapser`.

- Write a function called `spooner` which takes a pair of `String`s and exchanges their first letters. Use pattern matching- don’t use `head` or `tail`. For example:
  
  `spooner ("hi", "mom")` is ("mi","hom")

- Rewrite the `evenRange` function from class on 9/11 using guards. Call it `evenRange`. 
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:* you may want to define a separate function to use as part of this one.)