MA 151: Homework #3

due Tuesday September 26

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.

1. `snd (head (fst ("hello", 15)))`
2. `snd (fst [(1,2),(4,6)])`
3. `sum [4 * x | x <- [1..5]]`
4. `maximum [3 * length x | x <- ["fake", "it", "to", "make", "it"] ]`
5. `product [ x | x <- [1..20], mod x 4 == 0]`

Programming problems

Include a type signature for all functions you define. Make the type signatures polymorphic whenever possible.

- Use a list comprehension to create the list of all numbers less than 100 with exactly 6 divisors. (Use the `divs` function from class on 9/19.) Call this list `sixDivs`.

  This will be a function with no parameters, so just define it like: `sixDivs = [` *stuff here* `]`

- Use a list comprehension to create this list: [”a”,”ab”,”abc”, . . . , ”abcdefghijklmnopqrstuvwxyz”]. This function also will have no parameters. Call it `abcs`. (Hint: Use a list comprehension with `take` on the left of the bar, and a range of numbers on the right.)

- Use a list comprehension to create the list of all numbers up to 100, paired with their number of divisors. Call it `divPairs`. Your list should start like: [(1,1),(2,2),(3,2),(4,3),. . .]

- A number is a *perfect number* if it is the sum of each of its divisors (except for itself). For example 6 is a perfect number because the divisors of 6 are $\{1, 2, 3, 6\}$, and $1 + 2 + 3 = 6$. Also 28 is a perfect number (its divisors are $\{1, 2, 4, 7, 14, 28\}$).

  Use a list comprehension to create a the list of all perfect numbers. Call your list `perfects`. (*Hint:* perfect numbers are very rare, so you will have to run your program for a long time to get even the 4th one.) Fun fact: it is not known by mathematicians whether or not there are infinitely many perfect numbers.

- Write a function called `evenRange` using a list comprehension so that `evenRange x y` is the list of all numbers from x to y, but including only the even numbers. (You should include x and y if they are even.)

- Define a function called `notDivBy` which takes one `Int` parameter n and returns the infinite list of all positive numbers not divisible by n.
- Define a function called `increaser` which takes a list of `Int` and increases each element by 1. For example: `increaser [3,4,2,0,1]` is `[4,5,3,1,2]`
- Define a function called `listOfLengths` which takes a list of lists and returns a list of their lengths. For example:

  `listOfLengths [[1,4,5],[2,3],[],[1..10]]` is `[3,2,0,10]`
- Define a function called `isAPrefix` which takes two strings and returns a Bool which says whether or not the first string is a prefix of the second. Like `isAPrefix "hell" "hello"` is true, because "hello" starts with "hell". Or `isAPrefix "mom" "mother"` is false. (Hint: build a list of all the prefixes similar to what you did with `abcs`.)