Homework 2

Due: November 19

This exercise will be a brief introduction to the programming language Haskell with an emphasis on recursion; in the next homework we will use induction to prove the correctness of a recursive program.

Note: the *only* built-in functions/operators you will need are basic arithmetic (+, −, etc), basic comparators (==, <=, etc), and string concatenation (++). Do *not* use any other built-in functions.

1 Getting started

The starter code is linked from the Piazza post; copy it into a new Haskell environment in https://replit.com, or if you would prefer to compile and run the code locally then download the Haskell compiler (https://www.haskell.org/platform/).

Run the starter code - if everything is working, it should print out some placeholder values (a few 9999s and "placeholder"s). Note that the test code at the bottom is annotated with the correct outputs for each function; use that to help understand the function specifications since the descriptions below will be short.

Watch the video introduction to Haskell: https://mediaspace.illinois.edu/media/t/1_nwso5bu7.

2 Recursive remainder

Write a recursive remainder function `recRem`. You may assume that the second argument (the divisor) will always be positive (the given error will be thrown if it is not). Find the remainder by adding/subtracting y to x until x is in the correct range for a remainder. (Make sure your output follows the remainder definition from the textbook, even if the dividend is negative.)

3 Recursive string manipulation

- Write a recursive function `repeatHorizontal` which takes in an integer n and a string, and returns the concatenation of n copies of the string.
- Write a recursive function `alternate` which takes in an integer n and two strings, and returns a string which is the concatenation of n total alternating copies of the two strings. Do not test whether the input number is odd or even - design your recursion with only one recursive case and one base case.

- Use the `alternate` function to complete the definition of `repeatVertical`, which should be like `repeatHorizontal` except that it uses newlines (`''\n''`) to make a string repeat vertically when printed instead of horizontally. (`repeatVertical` does *not* need to be recursive; all the recursion can be handled within `alternate` which is already written. Optional: can you see how you could rewrite `repeatHorizontal` to also just delegate to `alternate`?)
- Finally, write the function `checkerboard` which produces an n by m checkered pattern from the given strings (see starter code annotations). Make sure to add an appropriate Haskell type signature.

4 What to turn in

You will be able to turn this in on PL starting in a few days. You will need a single file containing the template code with all the functions filled in (`recRem, repeatHorizontal, alternate, repeateVertical, checkerboard`)