Homework 3 Due March 25th 11:59 pm

Submission rules:

- You must submit a single .hs file with the following name: <firstName>-<lastName>-hw3.hs. Failure to do so will result in -10 points.
- You will lose 10 points if you put a module statement at the top of the file.
- You will lose 10 points for any import statements you have in your file and will automatically miss any problems you used an imported function on.
- If your file doesn't compile you will lose 10 points and miss any problems that were causing the compilation errors.
- You must use the skeleton file provided and must not alter any type signature. If you alter a type signature you will automatically miss that problem.

Problems

Problem 1 (5 pts)

Define a function *stutter* which takes a list of elements and returns a list where every element within the list is duplicated. For example:

- stutter [] => []
- stutter [1,2,3] => [1,1,2,2,3,3]
- stutter "Hello World" => "HHeellllloo WWoorrlldd"

Problem 2 (5 pts)

Define a function *compress* which takes in a list and eliminates all consecutive duplicate elements. For example:

- compress [] => []
- compress "HHeellllloo WWoorrlldd" => "Helo World"
- compress [1,2,2,3,3,3,4,4,4,4,1,1] => [1,2,3,4,1]

Problem 3 (10 pts)

Define a function *isSuffixOf* which takes two lists as arguments and returns True *iff* the first list is a suffix of the second. For example:

- isSuffixOf "bar" "" => False
- isSuffixOf "" "foo" => True
- isSuffixOf "bar" "foobar" => True
- isSuffixOf "bar" "foobarf" => False

Problem 4 (10 pts)

Define a function *isHarshad* which returns True *iff* a given number is Harshad number. A Harshad number is a number that is divisible by the sum of its digits. This function should be defined on the range [0..]. For example:

- isHarshad 7 => True
- isHarshad 18 => True
- isHarshad 11 => False

Problem 5 (5 pts)

Define a function *mc91* which calculates the McCarthy 91 function. This function should be defined on the range [0..] The function is defined as follows:

$$MC(n) = \begin{array}{ll} n - 10 & if n > 100 \\ MC(MC(n + 11)) & if n <= 100 \end{array}$$

For example:

- map mc91 [95..110] => [91,91,91,91,91,91,91,92,93,94,95,96,97,98,99,100]

Problem 6 (25 pts)

Write a function *goldbach* which when given an even number n returns a list of all pairs of primes which sum to n. Each pair in the list should be unique and the first prime in the pair should be smaller than the second. The list should also be in lexicographically sorted order. To write this function you will also need to write a function which tests primality. This function should be defined on the following range [0..]. For example:

goldbach 6 => [(3,3)]

goldbach 20 => [(3,17),(7,13)]

goldbach 40 => [(3,37),(11,29),(17,23)]

goldbach 102 => [(5,97),(13,89),(19,83),(23,79),(29,73),(31,71),(41,61),(43,59)]

Problem 7 (10 pts)

Define a function *increasing* which takes a list and returns True *iff* the list is in increasing sorted order and False otherwise. For example:

- increasing [1,2,3] => True
- increasing "ABCD" => True
- increasing [100,99..0] => False

Problem 8 (15 pts)

Define a function *select* which takes a *predicate* function and two lists and returns a list which contains elements from the second list where the predicate returned True when applied to the corresponding index in the first list. For example:

- select even [1..26] “abcdefghijklmnopqrstuvwxyz” => “bdfhjlnprtvxz”
- select (<= ‘g’) “abcdefghijklmnopqrstuvwxyz” [1..26] => [1,2,3,4,5,6,7]
- select odd [1..10] “hello world this is joe” => “hlowr”

Problem 9 (15 pts)

Define a function *prefixSum* which takes a list of numbers as its argument and returns a list of sums of all prefixes of the list. For example:

- prefixSum [1..10] => [1,3,6,10,15,21,28,36,45,55]
- prefixSum [2, 5] => [2,7]