Homework 2 Due February 28th 11:59 pm

Submission rules:

- All text answers must be given in Haskell comment underneath the problem header.
- You must submit a single .hs file with the following name: firstName-lastName-hw2.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.
- This means that any function which is causing compiler errors should be commented out. There will be no partial credit.
- 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.
- You will lose 10 points if you include a *main* function in your file.

Problems

Problem 1 (Exercise 4.6) (5 pts)

A positive integer is perfect if it equals the sum of its factors, excluding itself. Using a **list comprehension** and the function *factors*, define a function *perfects :: Int -> [Int]* that returns the list of all perfect numbers up to a given limit.

```
> perfects 500
[6,28,496]
```

Problem 2 (Exercise 4.9) (5 pts)

The scalar product of two lists of integers *xs* and *ys* of length n is given by the sum of the products of the corresponding integers:

$\sum_{i=0}^{n-1} xs_i * ys_i$

In a similar manner to *chisqr*, show how a **list comprehension** can be used to define a function *scalarproduct :: [Int] -> [Int] -> Int* that returns the scalar product of two lists.

For example:

```
> scalarproduct [1,2,3] [4,5,6]
32
```

Problem 3 (5 pts)

Define a function *topN* that is given an *Int* n and a list of pairs of *String*s and *Int*s and returns the *String*s with a corresponding *Int* greater than or equal to n in a list. You must use a **list comprehension** to define this function.

For example:

```
> topN 80 [("abe", 76), ("ben", 100), ("cal", 56), ("dan", 90)]
["ben", "dan"]
```

Problem 4 (15 pts)

Define a function *riffle* that takes two lists and interleaves their elements. For example:

```
> riffle [1,2,3] [4,5,6]
[1,4,2,5,3,6]
> riffle [1,2] [4,5,6]
[1,4,2,5,6]
> riffle [1,2,3] [4,5]
[1,4,2,5,3]
```

Problem 5 (Exercise 6.6) (2 pts each, 10 pts total)

Without looking at the definitions from the standard prelude, define the following library functions using **explicit recursion**:

1. Decide if all logical values in a list are *True*:

```
and' :: [Bool] -> Bool
```

2. Concatenate a list of lists:

```
concat' :: [[a]] -> [a]
```

3. Produce a list with n identical elements:

```
replicate' :: Int -> a -> [a]
```

4. Select the nth element of a list:

```
(!!!) :: [a] -> Int -> a
```

5. Decide if a value is an element of a list:

```
elem' :: Eq a => a -> [a] -> Bool
```

Problem 6 (15 pts)

Define a function *iotaIota* which takes in an *Int* n and returns a list of pairs of *Int*s such that:

```
> iotaIota 1
[(1,1)]
> iotaIota 2
[(1,1),(1,2),(2,1),(2,2)]
> iotaIota 3
[(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)]
```

You must define this **without** using a list comprehension.

Problem 7 (10 pts)

Define a function *matrixMap* which takes in a function f and a list of lists xss and applies f to each inner element of xss. You must define this function **without** explicit recursion or list comprehensions. Instead you must use the *map* function.

For example:

```
> matrixMap (+1) [[1,2,3],[4,5,6],[7,8,9]]
[[2,3,4],[5,6,7],[8,9,10]]
```

Problem 8 (Exercise 6.7) (10 pts)

Define a recursive function *merge :: Ord a => [a] -> [a] -> [a]* that merges two sorted lists to give a single sorted list. For example:

```
> merge [2,5,6] [1,3,4]
[1,2,3,4,5,6]
```

Problem 9 (Exercise 6.8) (10 pts)

Using *merge*, define a function *msort :: Ord a => [a] -> [a]* that implements merge sort, in which the empty list and singleton lists are already sorted, and any other list is sorted by merging together the two lists that result from sorting the two halves of the list separately.

Hint: first define a function *halve :: [a] -> ([a],[a])* that splits a list into two halves whose lengths differ by at most one.

Problem 10 (15 pts)

Define 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 must be unique and the first prime in the pair must be less than or equal to the second prime in the pair. The list should also be in lexicographically sorted order. This function should be defined on the 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)]
```