CS 403/503 Exam 2 **Spring 2017** **Name _____**

CS 403 ⇒ Score is based on your best 8 out of 10 problems.
CS 503 ⇒ Score is based on your best 9 out of 10 problems.
Extra credit will be awarded if you can solve additional problems correctly within the allotted time.

Write each function below using Haskell, and also show the type of each function.
You may show either the inferred type, or a declared type that suffices for the given examples.
Write any non-standard helper functions that you use, but you don't need to show their types.

You will likely need to use most of these standard functions and other Haskell features:
+, -, *, div, mod, ^, ==, <, <=, >, >=, not, &&, ||, head, tail, :, ++, reverse, !!, if-then-else, map, foldl, foldr, zip, filter, take, drop, let, while, pattern matching, anonymous functions, tuples, finite and infinite lists, lazy evaluation, list comprehensions, guards, user-defined data types.

1. (apply3 s t) takes triples s and t, applies each function in s to the corresponding value in t, and returns a triple with the results. Examples:
   apply3 (head, length, even) ("wxyz", [2,3,5,7], 11) returns ('w', 4, False).
   apply3 (odd, tail, reverse) (11, "wxyz", [2,3,5,7]) returns (True, "xyz", [7,5,3,2]).
   apply3 ((1+), not, (1–)) (9, True, 9) returns (10, False, –8).

   apply3 ::

2. (series s f n) returns a list of length n that starts with s, such that each successive value is obtained by applying function f to its predecessor. Examples:
   series 2 (^2) 5 returns [2,4,16,256,65536].
   series 3 (10+) 8 returns [3,13,23,33,43,53,63,73].
   series 5 (*2) 7 returns [5,10,20,40,80,160,320].
   series True not 6 returns [True, False, True, False, True, False].

   series ::

3. (mysqrt n) returns the truncated square root of non-negative integer n. Do not use the built-in "sqrt" or (^) functions. Example: mysqrt 31 returns 5. For full credit, your function should be efficient. Hint: Use a helper function that does binary search in the range 0 to n. For half credit, use a less efficient linear search.

   mysqrt ::

4. (factorize n) returns a list of all the prime factors whose product yields the non-negative integer n. Examples: factorize 360 returns [2,2,2,3,3,5], and factorize 361 returns [19,19]. For full credit, your function should be efficient. Hint: Use a helper function that checks possible factors in the range 2 to $\sqrt{n}$.

   factorize ::

5. (powerset s) takes a set s represented as a list, and returns a list of all subsets of s. Your function may arrange each list in any order, because the order of elements within a set does not matter. Examples:
   powerset [1,2,3] returns [[1,2,3], [1,2], [1,3], [1], [2,3], [2], [3], [ ]].
   powerset "abc" returns ["abc", "ab", "ac", "a", "bc", "b", "c", ""].

   powerset ::

6. Functions foldr' and foldl' are the same as built-in foldr and foldl, except that they only work on non-empty lists and lack an identity element parameter. When defining foldr' and foldl', do not call any of the built-in fold functions as helper functions. Examples:
   foldr' (^) [2,3,2] returns 2^(3^2) = 2^9 = 512.
   foldl' (^) [2,3,2] returns (2^3)^2 = 8^2 = 64.
   foldr' (>=) [False,True,False] returns False>=(True>=False) = False>=True = False.
   foldl' (>=) [False,True,False] returns (False>=True)>=False = False>=False = True.

   foldr' ::

   foldl' ::

Problems 7 and 8 refer to the following user-defined data type and variables. Observe that in this Tree definition, Leaf nodes contain values and Branch nodes contain lists of subtrees.

```
data Tree a = Leaf a | Branch [Tree a] deriving (Show)

t1 = Branch [Leaf 3, Branch [Leaf 4, Branch [Leaf 5, Leaf 6], Branch [ ]], Leaf 7]
t2 = Branch [Leaf 'a', Branch [Leaf 'b', Leaf 'c'], Leaf 'd', Branch [Leaf 'e', Leaf 'f'], Leaf 'g']
```

7. (mapall f t) applies function f to all the values in leaf nodes of tree t. Examples:
   mapall (^2) t1 returns
   Branch [Leaf 9, Branch [Leaf 16, Branch [Leaf 25, Leaf 36], Branch [ ]], Leaf 49].
   mapall (\x -> [x,x]) t2 returns
   Branch [Leaf “aa”, Branch [Leaf “bb”, Leaf “cc”], Leaf “dd”, Branch [Leaf “ee”, Leaf “ff”], Leaf “gg”].

   mapall ::

8. (level n t) returns a list of all the values in leaf nodes at depth n of tree t. Examples:
   level 1 t1 returns [3, 7].
   level 2 t1 returns [4].
   level 3 t1 returns [5, 6].
   level 1 t2 returns "adg".
   level 2 t2 returns "bcef".

   level ::

9. (table f) builds and returns an infinite 2-dimensional table by applying binary function f to the row and column numbers. Hint: use list comprehensions. Also, (subtable m n t) returns a finite table that consists of the first m rows and n columns of table t. Examples:
   subtable 4 6 (table (+)) returns [[0,1,2,3,4,5], [1,2,3,4,5,6], [2,3,4,5,6,7], [3,4,5,6,7,8]].
   subtable 7 3 (table (*)) returns [[0,0,0], [0,1,2], [0,2,4], [0,3,6], [0,4,8], [0,5,10], [0,6,12]].
   subtable 5 4 (table (^)) returns [[1,0,0,0], [1,1,1,1], [1,2,4,8], [1,3,9,27], [1,4,16,64]].

   table ::

   subtable ::

10. (column k t) returns the k^{th} column of 2-dimensional table t, and (diagonal t) returns the main diagonal of 2-dimensional table t. Examples (some use the table function from problem 9):
    column 2 ["abcde", "fghij", "klmno", "pqrst", "uvwxy"] returns "chmrw".
    diagonal ["abcde", "fghij", "klmno", "pqrst", "uvwxy"] returns "agmsy".
    column 3 (table (^)) returns an infinite list [0,1,8,27,64,125,...].
    diagonal (table (^)) returns an infinite list [1,1,4,27,256,3125,...].

    column ::

    diagonal ::