CS 403/503 Exam 2 Spring 2015 Name _____ Course ____

CS 403 ⇒ Score is based on your best 8 out of 12 problems.
CS 503 ⇒ Score is based on your best 10 out of 12 problems.
Extra credit will be awarded if you can solve additional problems.

Write each function below using Haskell, and also show its inferred type unless it's provided. To earn full credit, avoid using the "if-then-else" construct; instead use preferred Haskell constructs such as higher-order functions, pattern matching, list comprehensions, and/or guards. Also write any non-standard helper functions you choose to use, and show their inferred types as well.

1. convert (xs,ys,zs) returns a list of triples of the corresponding values from the lists xs, ys, zs. Example: convert ([1,2,3],[4,5,6],[7,8,9]) returns [(1,4,7),(2,5,8),(3,6,9)].

   convert ::

2. invert is the inverse function of convert. Example: invert [(1,4,7),(2,5,8),(3,6,9)] returns ([1,2,3],[4,5,6],[7,8,9]).

   invert ::

3. find v xs determines whether or not value v is an element of the ascending *infinite list* xs. Example: if evens = 0 : map (+2) evens, then find 8 evens returns True, and find 9 evens returns False.

   find ::

4. splitRange x y zs returns a pair of lists: the first list contains the values of zs that are in the range from x to y, and the second list contains the values of zs that are not in range from x to y. Example: splitRange 3 6 [0,2,4,6,8,10,9,7,5,3,1,0] returns ([4,6,5,3],[0,2,8,10,9,7,1,0]).

   splitRange ::

5. loop f s n applies function f repeatedly n times, beginning from the start value s. Example: loop tail [2,3,5,7,11,13,17,19,23,29] 4 returns [11,13,17,19,23,29].

   loop ::

6. add, mult, power, tower each has type Integer -> Integer -> Integer, with values restricted to non-negative integers. add, mult, power are the same as (+), (*), (^) respectively, and tower is repeated exponentiation. Example: tower 2 4 returns $2^{2^{2^2}} = 2^{2^4} = 2^{16} = 65536$. Assume that a helper function inc = (+1) exists, but otherwise your functions must not use any arithmetic operators directly.

7. extremes xs returns a list that contains the first and last values from each element of xs. Example: extremes ["abcd","efghij","klm"] returns ["ad","ej","km"].

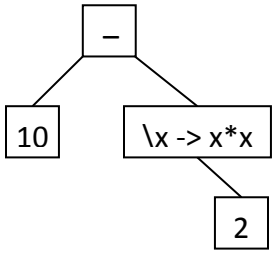
   extremes ::

8. curry3 and uncurry3 have the same purpose as predefined functions curry and uncurry, except they apply to functions with 3 parameter values rather than 2 parameter values.

   curry3 ::

   uncurry3 ::

9. eval e has type Expr -> Int, and evaluates the given expression e of type Expr. Also write a data type Expr that represents expressions built from Int values and unary and binary operators. Example: eval (BinaryOp (–) (Value 10) (UnaryOp (\x->x*x) (Value 2))) returns 6.



10. suffixes xs returns the list of all suffixes of xs. Example: suffixes [1,2,3,4] returns [[ ],[4],[3,4],[2,3,4],[1,2,3,4]].

    suffixes ::

11. prefixes xs returns the list of all prefixes of xs. Example: prefixes [1,2,3,4] returns [[ ],[1],[1,2],[1,2,3],[1,2,3,4]].

    prefixes ::

12. if' x y z always yields the same result as (if x then y else z). However, do not use if' as a helper function anywhere else on this exam. Also, suppose you are given a language that's identical to Haskell except it uses eager evaluation rather than lazy evaluation; provide and explain an example which shows that if' would not always behave the same as if.

    if' ::