ECS 140A: Summer 2024
Homework Assignment 3

Due Date: No later than Friday, July 19, 11:00pm PDT

For each of the following problems except problem 8, you are to provide two solutions: one using the pattern-matching techniques we discussed in class, and one not using those pattern-matching techniques (i.e., the ones we used in class this past week -- we'll talk about pattern-matching over lists in the coming week). For example, the two solutions for myappend might look like:

```
-- without pattern matching
myappend list1 list2
  | list1 == [] = list2
  | otherwise   = (head list1):(myappend (tail list1) list2)

-- with pattern matching
myappend_pm [] list2     = list2
myappend_pm (x:xs) list2 = x:(myappend_pm xs list2)
```

Note that I have ended the name of the pattern-matching solution with **"_pm"**. Please do the same for all the pattern-matching solutions that you submit for this assignment.

To solve problems 1-7, you may use only the following list operations -- ':', 'head', 'tail', 'null', and 'elem' . (Note that there are pattern matching equivalents for most of these.) Do not resort to just giving a new name to an existing Haskell function that already does what we want your function to do. So, for example

```
myappend inlist1 inlist2 = inlist1 ++ inlist2
```

wouldn't get you any points. Also, do not traverse any list more times than is necessary.

Please make sure you name your functions with the names provided below (with and without the _pm suffix), since your solutions will be autograded. Do not include any main function. Also, include type declarations with your functions. It will be good practice.

Submit all your solutions, including helper functions, to problems 1-8 as a single file named "hw3.hs".
For problems 1-7, grading will be on a 3-point scale for each

solution (7 problems x 2 solutions per problem x 3 points maximum per solution). Problem 8 is worth 15 points. Total = 57 points.

And now, here are your homework problems:

1) myremoveduplicates

```
myremoveduplicates "abacad" => "bcad"
myremoveduplicates [3,2,1,3,2,2,1,1] => [3,2,1]
```

2) myintersection

For this function, the order of the elements in the list returned by the function doesn't matter. Also, if the arguments to the function have duplicate elements in the list, then the result of the intersection is unspecified.

```
myintersection "abc" "bcd" => "bc"
myintersection [3,4,2,1] [5,4,1,6,2] => [4,2,1]
myintersection [] [1,2,3] => []
myintersection "abc" "" => ""
```

3) mynthtail

```
mynthtail 0 "abcd" => "abcd"
mynthtail 1 "abcd" => "bcd"
mynthtail 2 "abcd" => "cd"
mynthtail 3 "abcd" => "d"
mynthtail 4 "abcd" => ""
mynthtail 2 [1, 2, 3, 4] => [3,4]
mynthtail 4 [1, 2, 3, 4] => []
```

4) mylast

```
mylast "" => ""
mylast "b" => "b"
mylast "abcd" => "d"
mylast [1, 2, 3, 4] => [4]
mylast [] => []
```

5) myreverse

There’s a simple but inefficient solution to this problem, and a much more efficient solution. For full credit, provide the more efficient solution.

```
myreverse "" => ""
myreverse "abc" => "cba"
myreverse [1, 2, 3] => [3, 2, 1]
myreverse [] => []
```

6) myreplaceall

```
myreplaceall 3 7 [7,0,7,1,7,2,7] => [3,0,3,1,3,2,3]
myreplaceall 'x' 'a' "" => ""
myreplaceall 'x' 'a' "abacad" => "xbxcxd"
```

7) myordered

```
myordered [] => True
myordered [1] => True
myordered [1,2] => True
myordered [1,1] => True
myordered [2,1] => False
myordered "abcdefg" => True
myordered "ba" => False
```

8) computeFees

Using the same logic for computing student fees in homework 2, implement a single computeFees function in Haskell that calculates the correct fees depending on the student type, number of credits, financial aid, and etc. The input of computeFees is a single line of string from hw2.txt. It should return the correct fee calculation as an Int. You may need to create helper functions to achieve this. Note that the previous restrictions on using existing Haskell functions do not apply for this problem. You may provide either a pattern matching or a non pattern matching solution. You don’t need to provide both solutions for this problem.

```
computeFees “179250;Rick;Reichardt;53;14;Y;E;G;N” => 3450
computeFees “498545;William;Clinton;68;3;N;S” => 100
computeFees “412405;Keith;Utley;18;13;Y;S;G;Y;3500” => 0
```