159.202 Assignment 4

Deadline:	**Anytime before Saturday 26 Sept 2015**, time: mid night
Evaluation:	10 marks – which is 3% of your final grade
Late Submission:	5% per hour (or fraction of hour) it is late
Team:	The assignment can be done individually or in pairs.
Purpose:	Practice with Haskell tuples, lists and functions

This assignment consists of four exercises and is meant to help you practice with using tuples, lists and Haskell functions. You are expected to submit a single file named *a4.hs* consisting of

a) All authors name(s) and ID(s) (as comments at the top of the file)
b) Definitions, including types, **typeclasses** when possible, and good comments, for all functions is required for all the following exercises.

Exercise 1 [2 marks]

a) Write a function that takes as input a time as an integer number of seconds after midnight and converts it to the number of hours, minutes and seconds written as a triple: (`hours, minutes, seconds`). Call your function `time`.

For example, `time 50004` should evaluate to `(13, 53, 24).`

b) Consider the types:

```
type Point = (Int,Int)
type Line = (Int,Int,Int)
```

Write a function, `onLine`, to test if a point is on a given line.

For example: `onLine (2, -1) (2, 4, 1` should evaluate to `False` and
`onLine (-3,2) (2,-3,12)` should evaluate to `True.`

Exercise 2 [3 marks]

Consider the `Fraction` example from the Notes, page 114-115.

a) Define a function, `powFr,` to perform exponentiation for this type.

Example

`powFr (4,10) 3` should evaluate to $(4/10)^3$ and should behave like this:
`putStr (powFr (4,10))` will display: `8/125.`

b) Define a non-associative binary operator, %%, of biding power 3 that will return true if the two fractions have the numerators and denominators swapped.

For example (4, 5) %% (5, 4) should be true and (4, 5) %% (5, 3) should be false.

Exercise 3 [3 marks]

a) Define a recursive function `prod` that given two lists of integers will compute as its result the product of all the numbers in the first list that are divisible by the corresponding number in the second list.

For example:
`prod [11, 14, 22, 9] [2, 2, 4, 3]` is `126` (that is 14 * 9).
Your function should give a meaningful error if the lists have different sizes.

b) Implement a function called `magic` that takes a list of numeric values `nList` and returns a pair `(m,rest)` where `m` is the smallest element of `nList` and `rest` is the list obtained from `nList` after removing `m.` The elements of `rest` should be in the same order as the one in `nList.`

For example:
```
> magic [1,2,4,5,3]
(1,[2,4,5,3])
> magic [4,5,2,3]
(2,[4,5,3])
> magic [1]
(1,[])
```

c) Using recursion define a function `total` which takes a **list of lists** of numbers and returns the sum of all the integers in all of the lists. The sum of an empty list is 0.

For example:
```
> total [[1,2],[2,3],[3,4]]
15
> total [[],[1],[]]
1
> total []
0
```

Exercise 4 [3 marks]

a) Write a function `sOdd1` that takes a list `nums` of integers as argument and returns a list consisting of successors of odd elements from the list `nums`. Use recursions and patterns in your solution; no list comprehension and no higher order function is allowed.

For example:
```
> sOdd1 [2,3,18,5,22]
[4,6]
> sOdd1 [2]
[]
> sOdd1 [3,16,5]
[4,6]
```

b) Re-write the function, call it sOdd2, using recursions and guards this time; no list comprehension and no higher order function is allowed.
c) Re-write the function, call it sOdd3, using list comprehension and no recursion.

If you have any questions about this assignment, please ask the lecturer before its due time!

Submit your solution electronically using 159.202 Stream.

Important:

1. You can define and use other functions in order to solve a problem that asks you to define a function to perform a specific task.
2. Use only the material covered in lectures or in Notes (Stream); solutions using material not covered in lectures or in the above mentioned Notes will get 0 marks.
3. The assignment can be done individually or in teams of at most 2 students- send one solution file per team. All assignments authored by 3 or more students will get 0 marks.
4. Run your final version of *a4.hs* file on Albany computer labs and make sure it has no errors. Please note that if we cannot run your a4.hs script you will get 0 marks.
5. As sample solutions will be presented in lectures no extension will be possible.