**NCCU 程式語言PL, Spring 2021**

**Assignment 2 (Basics of Haskell Programming)**

**Due date: 13:00, March 23, 2021**

**(upload your program file,** 檔名**: PL-Assignment-2.hs)**

作業寫作前可以與同學討論，但要下筆時，請自己寫，不可抄襲，

請把握學習機會。一點一滴累積下來，進步指日可待，不要浪費學分。

**1. (Pattern matching, 20%)**

**(a)** Use pattern-matching with (:) and the *wildcard* pattern \_ to define a function, myButLast, that find the *last but one* element of a list. For examples;

myButLast :: [a] -> a

myButLast [1,2,3,4] = 3

myButLast ['a'[.](http://haskell.org/ghc/docs/latest/html/libraries/base/Prelude.html#v:.).'z'] = 'y'

**Note:** we assume that the input list has at least two elements.

**(b)** Use pattern-matching with (:) to define a function, rev2**,** that reverses all lists of length *2*, but leaves others unchanged. Ensure that your solution works for all lists --- that is, that the patterns you use are exhaustive. For examples:

rev2 [1, 2] = [2, 1], but rev2 [1, 2, 3] = [1, 2, 3].

You may use the standard Haskell function “reverse” in the body of rev2, but you **should not** use the “length” function to determine the length of the input parameter. You may also the “@” (*as-pattern*) to simplify your code.

f s@(x:xs) = x:s is a shorthand for f (x:xs)  = x:x:xs

**2. (List comprehension, 30%)**

(a) (10%) Using list comprehensions, define a function, countNeg, for counting the number of negative numbers in a list of numbers.

countNeg :: [Int] -> Int

>countNeg [1, -2, 3, -5] = 2

(b) (10%) Define *xn* using a list comprehension. Name the function as raise:

raise :: Int -> Int -> Int

>raise 2 4= 16

(c)(10%) **Pascal's triangle** is a triangle of numbers

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

.............

computed as follows: (1) The first row just contains a 1.

(2) The following rows are computed by adding together adjacent numbers in the row above, and adding a 1 at the beginning and at the end.

Write a function, pascal, using list comprehension and ++, which maps a positive integer *n* to the *nth* row of Pascal numbers.

For example, >pascal 5 = [1, 4, 6, 4, 1].

Hint: define an auxiliary function pairs which construct pairs from two consecutive integers in a list.

**3. (Higher-order functions and list comprehension 30%)**

(a) (10%) Study the code fragment below and identify the operation it provides. Then rewrite it using mapand filter, and name it as **q1f1a.**

q1f1 :: [Int] -> [Int]

q1f1 [] = []

q1f1 (x:xs) | x < 3 = q1f1 xs

| x > 10 = q1f1 xs

| otherwise = x\*3 : q1f1 xs

(b) (10%) Now rewrite q1f1using *list comprehensions and name it as* **q1f1b***.*

(c) (10%) Express the comprehension

[f x | x <- xs, p x]

as a function using the functions **map** and **filter.** Call the function **compre:**

compre :: [a]->(a->b)->(a->Bool)->[b]

compre xs f p = … --using map and filter

**4. (Higher-order function, 10%)** We use lists to represent sets. Your task is to define a function, subsets, that receives a list as a set and returns the set of all subsets of the input set. For example:  
 subsets :: [Int] -> [[Int]]  
 subsets [] = [ [] ]  
 >subsets [1,2,3] =  
 [ [], [1], [2], [3],[1,2] [1,3], [2,3], [1,2,3] ] **–順序無關**

**5.(Lambda in other PL’s, 10%) 閱讀** [**https://medium.com/@afinlay/lambda-expressions-in-java-python-c-c-8cdbca5a5e8b**](https://medium.com/@afinlay/lambda-expressions-in-java-python-c-c-8cdbca5a5e8b)，接著用你熟悉的另一種程式語言，找出它的lambda的語法以及filter 高階函數的使用方式，應用filter寫出一個範例。使用Python的範例如下：

fruit = ["Apple", "Banana", "Pear", "Apricot", "Orange"]

filter\_object = filter(…??..., fruit)