THE UNIVERSITY OF HONG KONG

COMP3258: FUNCTIONAL PROGRAMMING

Assignment 1

Deadline: 23:59, Oct 6, 2017 (HKT)

Problem 1. (10 pts.) Implement a recursive function `combinations :: Int -> Int -> Int`, which takes integer `k` and `n`, and returns the number of k-combinations selected from `n` elements.

Notice:

- $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$
- Assume all the inputs are in the range $0 \leq n \leq 20$ and $0 \leq k \leq n$

Expected running results:

```
*Main> combinations 0 0
1
*Main> combinations 1 3
3
*Main> combinations 5 8
56
*Main> combinations 4 10
210
```

Problem 2. (15 pts.) There is a mapping from integers to the corresponding column titles as appear in an Excel sheet.

```
1 -> A
2 -> B
3 -> C
...
26 -> Z
27 -> AA
28 -> AB
...
```

Implement a recursive function `intToColumn :: Int -> String`, which returns the column title as string for a given integer.

Notice:

- Assume the input is a non-negative integer
- If the input number is zero, return an empty string `""`
- The output string should only contain upper-case letters (`'A'` to `'Z'`)

Expected running results:

```
*Main> intToColumn 1
"A"
*Main> intToColumn 12
"L"
*Main> intToColumn 28
"AB"
*Main> intToColumn 10000
"NTP"
```

Problem 3. (10 pts.) You got a phonebook which stores many phone numbers. However, the numbers have several different formats. A number `12345678` may be stored as:

- `12345678`
- `1234 5678`
- `1234-5678`
- `12-34 56-78`

and so on. In general, there may be hyphens `'-'` and spaces `' '` together with the digits in a phone number string.

Please write a function `lookupPhoneNumber :: [String] -> String -> [String]`, which takes a phonebook as a list of strings, and a prefix of phone number as a single string, return all numbers that start with the prefix.

Notice:

- Assume all inputs are valid as below:
  - All numbers in the phonebook contain only hyphens `'-'`, spaces `' '`, and digits from `'0'` to `'9'`
  - The prefix contains only digits
- If the prefix is empty, return all the numbers in the phonebook

- Numbers which are shorter than the prefix should not be returned
- The numbers returned by your function should be well formatted. That is, every number only contains digits, without hyphens and spaces

Expected running results:

```
*Main> lookupPhoneNumber ["1234-5678"] "123"
["12345678"]
*Main> lookupPhoneNumber [" 1 2-34-5678"] "123"
["12345678"]
*Main> lookupPhoneNumber [" 1 2-34-5678", "1-230-0123"] "123"
["12345678","12300123"]
*Main> lookupPhoneNumber ["123"] "123456"
[]
*Main> lookupPhoneNumber ["123-45 6-78"] ""
["12345678"]
```

Problem 4. (20 pts.) In Haskell, there are many filter operations on lists. For example `filter even [1,2,3,4] = [2, 4]`. Some filter patterns are so common so Haskell provides functions for them. For example, `takeWhile :: (Int -> Bool) -> [Int] -> [Int]` will take the elements of the list until the predicator fails. `takeWhile even [2,4,3,6] = [2,4]`. There is also `dropWhile` that drops the elements until the predicator fails. Let's make our own filter from it!

- (10 pts.) Use `foldl` or `foldr` to implement our own `myFilter :: (Int -> Bool) -> [Int] -> [Int]` function, which has the same behavior as `filter`.
- (10 pts.) Implement `takeFromUntil :: (Int -> Bool) -> (Int -> Bool) -> [Int] -> [Int]` that takes two predicator inputs, take elements from where the first predicator holds, until the second predicator holds.

Notice

- For `takeFromUntil`, if the start point satisfies the second predicator, the function stops immediately with result `[]`.

Expected running results:

```
*Main> myFilter even [1,2,3,4]
[2,4]
*Main> myFilter (\x -> x `mod` 3 == 0) [1,2,3,4]
[3]
*Main> takeFromUntil odd even [6,1,3,2,4,5]
[1,3]
*Main> takeFromUntil odd odd [1,2,3,4]
```

```
[]
*Main> takeFromUntil odd even [2,4,6]
[]
*Main> takeFromUntil (\x -> x == 1) (\x -> x == 3) [1,2,3]
[1, 2]
```

Problem 5. (10 pts.) The stop-vowel-stop pattern is quite common in English. For example, given `vowels = "aeiou"`, `stops="ctdg"`, some valid English words are "cat", "dog", etc. Use list comprehension to implement `words :: String -> String -> [String] -> [String]` that takes `vowers`, `stops`, and `dictionary` as input, return all stop-vowel-stop words that are valid in the dictionary.

Expected running results:

```
*Main> words "ai" "dgp" ["haskell", "dig", "world", "gap"]
["dig", "gap"]
*Main> words "u" "cpt" ["cup", "put"]
["cup", "put"]
*Main> words "aei" "cp" ["hello", "cat"]
[]
```

Problem 6. (10 pts.) Sorting is really handy in Haskell. Run `sort [1,4,3,2]` and you can get `[1,2,3,4]` immediately. Let's make some interesting permutation from it! Implement `wave :: [Int] -> [Int]`, which returns a list that:

- The first element is the largest one.
- The second element is the smallest one.
- The third element is the second to largest (could be the same) one.
- The fourth element is the second to smallest (could be the same)one.
- ...

Notice

- The return list should have the same length as the input, and contain all the elements in the input list.
- Duplication is allowed.

Expected running results:

```
*Main> wave [1,2,3,4]
[4,1,3,2]
*Main> wave [1,1,2,2]
[2,1,2,1]
*Main> wave [3,2,1]
```

```
[3,1,2]
*Main> wave []
[]
```

Problem 7. (20 pts.) Do you remember the good friend you had when you learned basic mathematics: the calculator! Having all you got from this course so far, you are ready to implement a toy calculator. Given the signature `calculator :: String -> Int`, implement a calculator based on following rules:

- The input is a basic mathematical expression containing only numbers and operators, e.g. `1 + 2`.
- Calculate the input by order, instead of precedence, e.g. `1 + 2 * 5 = 15`.
- All numbers are non-negative integers (`Int`).
- The calculator supports operator: `+`, `-`, `*`, `/`.
- Use `div` for division, e.g. `5 div 4 = 1`, `4 div 5 = 0`.
- There could be blanks between numbers and operators, e.g. `1  + 3+4`.
- No blanks between numbers, e.g. `1  2 + 3` is invalid.
- The input is not broken, e.g. `1 +- 2`, `1 +`, or `- 1` is invalid.
- Return `-1` if there is any error, e.g. division by zero, and ignore the rest input.
- No test case will cause integer overflow.

Expected running results:

```
*Main> calculator "1 + 2 + 3"
6
*Main> calculator "4 + 5 * 2 / 0 + 1"
-1
*Main> calculator "1*2*3/4"
1
*Main> calculator ""
0
```

Code style and submission (5 pts.)

All functions should be implemented in a single Haskell file, named as A1_XXX.hs, with XXX replaced by your UID. Your code should be well-written (e.g. proper indentation, names, and type annotations) and documented. Please submit your solution on Moodle before the deadline.

Notice: there are cases that students cannot upload a Haskell file on Moodle. Then please compress it into A1_XXX.zip, which contains only one file A1_XXX.hs.

Plagiarism

Please do this assignment on your own; if, for a small part of an exercise, you use something from the Internet or were advised by your classmate, please mark and attribute the source in a comment. Do not use publicly accessible code sharing websites for your assignment to avoid being suspected of plagiarism.