The University of Hong Kong

COMP3258: Functional Programming

Assignment 1

Deadline: 23:59, March 3, 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.) Implement a recursive function `position :: Eq a => a -> [a] -> Int`. It should return the index of the first element which equals to the query value in the given list. If there is no such element, return `-1`.

Notice:

- Indexes start from `0`

Expected running results:

```
*Main> position 1 [1,2,3]
0
*Main> position 2 [1,2,3]
1
*Main> position 3 [1,2,3]
2
*Main> position 4 [1,2,3]
-1
```

Problem 3. (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 4. (15 pts.) Continuing with the previous question, implement a recursive function `columnToInt :: String -> Int`. It should return the corresponding integer as the column number for a given column title.

Notice:

- Return zero if the input string is empty
- Assume all other input strings contain only upper-case letters

Expected running results:

```
*Main> columnToInt "A"
1
*Main> columnToInt "ZZ"
702
*Main> columnToInt "CV"
100
*Main> columnToInt "AB"
28
*Main> columnToInt "HELLO"
3752127
```

Problem 5. (10 pts.) To test `intToColumn` and `columnToInt` by QuickCheck, finish the property `propColumnNumber` below. The property should be that, if you convert an integer to a column title by `intToColumn`, then function `columnToInt` can convert the title back to the initial integer.

```
propColumnNumber :: Int -> Property
propColumnNumber x = (???) ==> ???
```

Notice:

- Fill all the `???`, and test your functions

Problem 6. (15 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 7. (15 pts.) Implement a function `permutations :: [a] -> [[a]]`, which returns all permutations of the input list.

Notice:

- The order of the returned permutations does not matter (that is, for input `[1,2]`, result `[[1,2],[2,1]]` and `[[2,1],[1,2]]` are both correct

Expected running results:

```
*Main> permutations []
[[]]
```

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

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.

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.