Homework Assignment 1 – Solutions

September 29, 2013

1 Deriving adjectives from nouns and suffixes

ghci 1> let {nouns, suffixes, adjectives :: [String];
    nouns = ["color","faith","use"];
    suffixes = ["ful","less"];
    adjectives = [x ++ y | x ← nouns, y ← suffixes]}

ghci 2> nouns
["color","faith","use"]

ghci 3> suffixes
["ful","less"]

ghci 4> adjectives
["colorful","colorless","faithful","faithless","useful","useless"]

2 Split a text on whitespace, count words

ghci 5> let text = "Pierre Vinken , 61 years old , will join the board " ++
    "as a nonexecutive director Nov. 29 . Mr. Vinken " ++
    "is chairman of Elsevier N.V. , the Dutch publishing group ."

ghci 6> text
"Pierre Vinken , 61 years old , will join the board as a nonexecutive director Nov. 29 . Mr. Vinken is chairman of Elsevier N.V. , the Dutch publishing group ."

A. Split this text on whitespace and print the resulting list of words:
B. Determine the length of this list, i.e., how many words we have in the text:

ghci 9> length text_words
31

3 Extract lengths, determine shortest and longest words, determine average word length

A. Generate the list of lengths for all the words:

ghci 10> let length_words = map length text_words

ghci 11> length_words
[6, 6, 1, 2, 5, 3, 1, 4, 4, 3, 5, 2, 1, 12, 8, 4, 2, 1, 3, 6, 2, 8, 2, 8, 4, 1, 3, 5, 10, 5, 1]

B. Identify the minimum and maximum length in the list:

ghci 12> let min_length = minimum length_words

ghci 13> min_length
1

ghci 14> let max_length = maximum length_words

ghci 15> max_length
12

C. List all the words that have the minimum length and also all the words that have the maximum length:
D. Determine the average word length in this text:

```
ghci 16> [word | word ← text_words, length word ≡ min_length]
["","","","a","","","."]
```

```
ghci 17> [word | word ← text_words, length word ≡ max_length]
["nonexecutive"]
```

4 Write a function that behaves like `drop`

```
ghci 21> let { drop' :: Int → [a] → [a];
    drop' _ [ ] = [ ];
    drop' n xs@(y : ys)
      | n ≤ 0 = xs
      | n > 0 = drop' (n - 1) ys }
```

```
ghci 22> drop' 5 "Supercalifragilisticexpialidocious"
"califragilisticexpialidocious"
```

```
ghci 23> drop' 9 "Supercalifragilisticexpialidocious"
"fragilisticexpialidocious"
```

```
ghci 24> drop' 0 "Supercalifragilisticexpialidocious"
"Supercalifragilisticexpialidocious"
```

```
ghci 25> drop' (-3) "Supercalifragilisticexpialidocious"
"Supercalifragilisticexpialidocious"
```
And this is an alternative definition:

```haskell
ghci 27> let {drop'' :: Int → [a] → [a];
           drop'' n xs =
               if n ≤ 0 ∨ null xs
               then xs
               else drop'' (n - 1) (tail xs)}
```

```haskell
ghci 28> drop'' 5 "Supercalifragilisticexpialidocious"
   "califragilisticexpialidocious"
```

```haskell
ghci 29> drop'' 9 "Supercalifragilisticexpialidocious"
   "fragilisticexpialidocious"
```

```haskell
ghci 30> drop'' 0 "Supercalifragilisticexpialidocious"
   "Supercalifragilisticexpialidocious"
```

```haskell
ghci 31> drop'' (-3) "Supercalifragilisticexpialidocious"
   "Supercalifragilisticexpialidocious"
```

```haskell
ghci 32> drop'' 4 ""
   ""
```