Portfolio element – Haskell

Unit	Programming languages: principles and design (6G6Z1110) Programming languages – SE frameworks (6G6Z1115)
Lecturer	Rob Frampton
Week	11
Portfolio element	Haskell (15% of coursework)

Introduction

In this assignment, you will implement a Haskell program which reads some text from a file and displays some word statistics:

1) The total number of words in the text.
2) The total number of the top 20 most commonly used English words that appears in the text according to the Oxford English Corpus (OEC) rank. See the list in Figure 3.
3) A histogram of the top 20 most frequent words in the text excluding common words.

A program template is provided for you (available on Moodle):

```
<your code here>

text = "It was the best of times, it was the worst of times, it was the age of wisdom,
it was the age of foolishness, it was the epoch of belief, it was the epoch of
incredulity, it was the season of Light, it was the season of Darkness, it was the
spring of hope, it was the winter of despair, we had everything before us, we had
nothing before us, we were all going direct to Heaven, we were all going direct the
other way--in short, the period was so far like the present period, that some of its
noisiest authorities insisted on its being received, for good or for evil, in the
superlative degree of comparison only.\nThere were a king with a large jaw and a queen
with a plain face, on the throne of England; there were a king with a large jaw and a
queen with a fair face, on the throne of France. In both countries it was clearer than
crystal to the lords of the State preserves of loaves and fishes, that things in general
were settled for ever."

main = do
  let wordlist = toWordList text
  putStrLn "Report:"
  putStrLn ("\t" ++ (show $ length wordlist) ++ " words")
  putStrLn ("\t" ++ (show $ countCommonWords wordlist) ++ " common words")
  putStrLn "\nHistogram of the most frequent words (excluding common words):\n"
  putStr $ makeHistogram $ sortWords $ countWords $ dropCommonWords $ wordlist
```

Figure 1. "main" function of the program (file available on Moodle)

When complete, the expected output of the program is:

```
Report:
       185 words
       73 common words
Histogram of the most frequent words (excluding common words):
************ -> was
***** -> were
**** -> we
** -> us
** -> times
** -> throne
```

```
** -> there
** -> season
** -> queen
** -> period
** -> large
** -> king
** -> jaw
** -> its
** -> had
** -> going
** -> face
** -> epoch
** -> direct
** -> before
```

Figure 2. Expected output from the completed assignment

the	and	have	not	as
be	a	I	on	you
to	in	it	with	do
of	that	for	he	at

Figure 3. List of the top 20 most frequently used words in English according to the OEC rank.

Assignment

You must complete the program shown in Figure 1 by implementing the missing functions. Your complete program should execute as shown in Figure 2. Table 1 shows the list of missing functions you should implement, along with brief descriptions and examples of use. You can use those examples to test the output of your functions before you add them to your program.

Your program must contain the implementation of the functions listed on Table 1, at least. You may make additional functions if you wish. Modifications of the "`do`" block (Figure 1) are not permitted.

Table 1. List of the missing functions that must be implemented

Function name	*Brief description*	*Function call example*
`toWordList`	Takes a string, lowercases it, drops any character that is not a letter or a space, and returns a list with the words in the string.	`> toWordList "Hello, World! HELLO!! :-)"` `["hello","hello","world"]`
`countCommonWords`	Takes a list of strings and returns the number of times the 20 most frequently used English words appears in the list.	`> countCommonWords ["the","planet","of","the","apes"]` `3`
`dropCommonWords`	Takes a list of strings and drops any word that is within the top 20 most commonly used in English. Returns a list of strings without those words.	`> dropCommonWords ["the","planet","of","the","apes"]` `["planet","apes"]`
`countWords`	Takes a list of strings and returns a list. Each element of the returned list is a tuple which contains a string (a word) and an integer (representing the number of times the word appears in the	`> countWords ["friend","she","she"]` `[("friend",1),("she",2)]`

	text).	
`sortWords`	Sorts words by their frequency in descending order. It takes and returns a list of tuples. Each element of the tuple consists of one string (the word) and one integer (its frequency).	`> sortWords [("friend",1),("she",2)]` `[("she",2),("friend",1)]`
`makeHistogramRow`	Makes a string representing histogram row using asterisks. It takes a tuple (string, integer) and returns a string consisting of a number of asterisks (the second element of the tuple), the string `" -> "`, the word (the first element of the tuple), and finally a newline (`"\n"`).	`> makeHistogramRow ("test", 10)` `"********** -> test\n"`
`makeHistogram`	Makes a histogram using asterisks. It takes a list of tuples (string, integer) and returns a string which contains the histogram. This function should use the `makeHistogramRow` method to generate the string.	`> makeHistogram [("her",4),("she",2),("friend",1)]` `"**** -> her\n** -> she\n* -> friend\n"`

Submission

You must submit through Moodle a unique file named "*words.hs*" which contains the complete program including the implementation of the missing functions listed in Table 1, the code in Figure 1 without any modification, and any additional function you needed to implement in order for your program to be fully functional (as shown in Figure 2). Submission link is available in Week 11 section on Moodle.