Logic and Programming
Homework 4

Péter Mekis
Department of Logic, ELTE Budapest

Deadlines: October 16 10:00 am

- Define *any three of the following fifteen functions* in Haskell. Defining more than three is appreciated, but not necessary. Some of the exercises are follow-ups to others; it may be a good idea to choose them together.
- Also, define *three functions that aren't in this list*, based on your ideas, preferably connected to your main field of interest.
- Use recursion in all of your definitions. Don't use list comprehension (you can use list definitions of the form `[1..10]`, `['a'..'z']` etc.) If a Haskell library has a built-in solution for an exercise, don't use it. If you need syntax that wasn't covered in the class, google it. If the description doesn't cover all cases, be creative!
- If you use code that was created by someone else, please indicate it.
- The exercises range from the more elementary to the more advanced. Choose those that are at your level. Have fun! :)

Elementary recursion

1. **Type** `String -> Char`

   Description Finds and returns the middle element of the list, if there is one. Otherwise it returns an exclamation mark.

   Example

   ```
   > middlechar "abc"
   'b'
   > middlechar "abcd"
   '!'
   ```

2. **Type** `[String] -> String`

   Description Concatenates a list of strings.

   Example

   ```
   > concatenate ["abc","d","ef"]
   "abcdef"
   ```

3. **Type** `String -> Char -> Char`

   Description Finds the character next to the first occurrence of a character in a string. If there's none, it returns an exclamation mark.

   Example

   ```
   > nextto "Gottlob Frege" 'o'
   't'
   > nextto "abc" 'd'
   '!'
   ```

4. **Type** `String -> [String]`

   Description Slices up a string into substrings that consist of a single character

   Example

   ```
   > slice "Trump"
   ["T","r","u","m","p"]
   ```

5. **Type** `String -> (String, String)`

   Description Separates the vowels and the consonants of a word. Neglects any other character.

   Example

   ```
   > separate "Donald Trump"
   ("oau","DnldTrmp")
   ```

Intermediate recursion

1. **Type** `String -> String`

   Description Reduces a string so that it keeps only the first occurrences of every character.

   Example

   ```
   > reducestring "aaargh"
   "argh"
   > reducestring "Gottlob Frege"
   "Gotlb Freg"
   ```

2. **Type** `String -> [String]`

   Description Creates a list with all substrings of a string. (The examples represent two different approaches.)

   Example

   ```
   > substrings "abc"
   ["", "a", "b", "c", "ab", "bc", "abc"]
   > substrings' "abc"
   ["", "a", "ab", "abc", "b", "bc", "c"]
   ```

3. **Type** `String -> [String]`

   Description Splits a string at the occurrences of a given character.

   Example

   ```
   > split "my body is walking in space" ' '
   ["my","body","is","walking","in","space"]
   > split "ab.c.de.fgh"
   ["ab","c","de","fgh"]
   ```

4. **Type** `[(Int, Char)] -> String`

   Description Creates a string from a list of ordered pairs where the second member is a character and the first member is the number of its consecutive occurrences.

   Example

   ```
   > expand [(1,'a'),(2,'b'),(3,'c')]
   "abbccc"
   ```

5. **Type** `String -> Int -> [String]`

   Description Lists all the words of a given length of an alphabet in alphabetical order. (We attacked this problem in the October 9 session, it will suffice to just debug the (almost properly working) code we put together.)

Example

```
> wordlist "01" 2
["00","01","10","11"]
> wordlist "abc" 3
["aaa","aab","aac","aba","abb","abc","aca","acb","acc",
"baa","bab","bac","bba","bbb","bbc","bca","bcb","bcc",
"caa","cab","cac","cba","cbb","cbc","cca","ccb","ccc"]
```

Advanced recursion

1. **Type** `[Int] -> [Int]`

   Description Sorts a list of integers using the bubble sort algorithm. For further details, cf.

   `https://en.wikipedia.org/wiki/Bubble_sort`

   Example

   ```
   > bubblesort [3,2,4,1]
   [1,2,3,4]
   ```

2. **Type** `[Int] -> [Int]`

   Description Sorts a list of integers using the quicksort algorithm. For further details, cf.

   `https://en.wikipedia.org/wiki/Quicksort`

   Example

   ```
   > quicksort [3,2,4,1]
   [1,2,3,4]
   ```

3. **Type** `String -> String -> String -> String`

   Description Substitutes $string_2$ for the first occurrence of $string_1$ in $string_3$ if there is such an uccurrence; otherwise just returns $string_3$.

   Example

   ```
   > replace "is" "was" "a rose is a rose is a rose is a rose"
   "a rose was a rose is a rose is a rose"
   ```

4. **Type** `String -> String -> Int`

   Description Returns the Gödel number of a string. Eg., "def"'s Gödel number is $2^4 \cdot 3^5 \cdot 5^6$. In general, if a string's nth character is the mth member of the alphabet, the Gödel number will contain ${p_n}^m$, where p_n is the nth prime number. Cf.

   `https://en.wikipedia.org/wiki/G%C3%B6del_numbering#G.C3.B6del.27s_encoding`

   Example

   ```
   > encode "def" "a..z"
   60750000
   ```

5. **Type** `Int -> String -> String`

   Description Decodes a word from its Gödel number.

   Example

   ```
   > decode 75600 "a..z"
   "dcba"
   ```