Programming in Haskell – Midterm Exam

UNIZG FER, 2016/2017

Note: Define each function with the exact name and the type specified. You can (and in most cases you should) define each function using a number of simpler functions. Provide a type signature above each function definition and comment the function above the type signature. Unless said otherwise, a function may not cause runtime errors and must be defined for all of its input values. Use the `error` function for cases in which a function should terminate with an error message.

Each problem is worth a certain number of points. The points are given at the beginning of each problem, and are scaled to 10 upon grading.

You need at least *5 points* (after scaling) to pass this midterm exam. You have one hour for solving this exam. You are to solve it on your laptop and may use the Internet, save for communicating with one another, or third parties.

1. *(2 pts)* Given a sentence, define a function called `capitalise` which returns the same sentence with all words capitalised except the ones from a given list.

```
capitalise :: String -> [String] -> String
capitalise "this is a sentence." ["a", "is"] ⇒ "This is a Sentence."
```

2. *(3 pts)* Given a list of words, remove from the list all those which contain four or more vowels and those which have same letter appearing twice (or more) in a row. In addition, any word containing numbers should have the numbers removed. Note that the number removal should happen before any other operations so that the subsequent operations can remove the word if necessary.

```
weirdFilter :: [String] -> [String]
weirdFilter ["abc", "bananae", "fuzzy", "c1c2"] ⇒ ["abc"]
```

3. *(2 pts)* Write a function called `gcd'` which calculates the greatest common divisor of two numbers `a, b`. The function should behave the same as the built-in `gcd` function. Here's it's pseudocode to help you. (you can of course use any other algorithm)

```
gcd' :: Int -> Int -> Int
gcd' 120 30 ⇒ 30
gcd' 120 31 ⇒ 1
```

```
while(b > 0)
    (a, b) = (b, a mod b)
return a
```