Homework 1

1. Install the Glasgow Haskell Compiler or the Haskell Platform and create a new file called `hw1.hs` which includes all your code.

   Be sure to test your file with GHCi or by compiling it with ghc.

   (5 points)

2. Write a function

   ```
   citeAuthor :: String -> String -> String
   ```

   which puts first name and last name in reverse order:

   ```
   citeAuthor "Herman" "Melville" -- -> "Melville, Herman"
   ```

   (5 points)

3. Write a function

   ```
   initials :: String -> String -> String
   ```

   which returns the initials of the provided first name and last name:

   ```
   initials "Herman" "Melville" -- -> "H.M."
   ```

   (5 points)

4. Suppose that we represent books `(author, title, year)` as tuples `(String, String, Int)`.

   Write a function

   ```
   title :: (String, String, Int) -> String
   ```

   which returns the title of a book.

   ```
   title ("Herman Melville", "Moby Dick", 1851) -- -> "Moby Dick"
   ```

   Hint: use pattern matching

   (5 points)

5. Write a function

   ```
   citeBook :: (String, String, Int) -> String
   ```

   which returns a citation in the format `title (author, year)`

   ```
   citeBook ("Herman Melville", "Moby Dick", 1851) -- -> "Moby Dick (Herman Melville, 1851)"
   ```

(10 points)

6. Write a function

   ```
   bibliography_rec :: [(String, String, Int)] -> String
   ```

   which returns a string containing all the books as citations in the form returned by `citeBook` in part 5, separated by newlines. Use recursion and your previous `citeBook` function to build up the result.

   (10 points)

7. Write a function

   ```
   bibliography_fold :: [(String, String, Int)] -> String
   ```

   which does the same as `bibliography_rec` but instead of using recursion, it uses `foldl` to build up the string.

   (10 points)

8. Write a function

   ```
   averageYear :: [(String, String, Int)] -> Int
   ```

   which returns the average publication year of the provided books.

   ```
   averageYear [("","",1),("","",3)] -- -> 2
   ```

   Hint:
   `div` performs integer division, `sum` computes the sum of a list of numbers, and `map` takes a function and a list and returns a list with all elements mapped by the function

   (15 points)

9. Write a function

   ```
   references :: String -> Int
   ```

   which takes a text with references in the format `[n]` and returns the total number of references.

   ```
   txt :: String
   txt = "[1] and [2] both feature characters who will do whatever it takes to " ++
         "get to their goal, and in the end the thing they want the most ends " ++
         "up destroying them.  In case of [2] this is a whale...
   references txt -- -> 3
   ```

   You can assume that the input text has at most one digit in between the square brackets and nothing else, that every reference will be followed by a space, and that square brackets are never used for anything but references.

   Hints:
   `words` splits a string into a list of words,

`filter` selects only certain elements in a list
`length` returns the length of a list

(15 points)

10. Write a function

```
citeText :: [(String, String, Int)] -> String -> String
```

which takes a list of books and a text with references in the form `[n]` and returns a text with all references replaced by a citation of the n'th book using the `citeBook` function from problem 5.

```
let gatsby = ("F. Scott Fitzgerald", "The Great Gatsby", 1925)
let moby = ("Herman Melville", "Moby Dick", 1851)
citeText [gatsby, moby] txt
-- "The Great Gatsby (F. Scott Fitzgerald, 1925) and Moby Dick (Herman Melville,
1851) both feature..."
```

You can assume that the input text has at most one digit in between the square brackets and nothing else, that every reference will be followed by a space, and that square brackets are never used for anything but references.

Hints: `unwords` is the opposite of `words`

(20 points)