Synopsis Lab 7

7.1 Tasks

1. Skim lectures #4 and #5. Test the following functions: `prod`, `flatten`, `zip`, `unzip`, `df` and list comprehensions. Tip: check folder `src` folder for (some of) the examples in the lectures.

2. (Haskell) Write a function `safetail`, which is similar to `tail` for non-empty lists and which returns `[]` if called on an empty list.

3. (Haskell) Write a function `palindromeL` which tests if a list in palindromic *in letters* (spaces will be ignored). E.g.:

```
Main> palindromeL ["a man", "a plan", "a canal", "panama"]
True
```

4. (Haskell) Write a function `decimate`, having 2 arguments: a list `L` and a number `n`, which eliminates every nth element of list `L`. E.g.:

```
Main> decimate [1..16] 5
[1,2,3,4,6,7,8,9,11,12,13,14,16]
```

5. (Haskell) Write a function called `addBigs` which adds two big numbers given as lists of digits. E.g.:

```
Main> addBigs [1,3,9]
              [2,2,2]
[3,6,1]
```

6. (ML) Write a function called `bf` which lists all nodes of a dag in a breadth-first manner. It has 3 parameters: a list with only one element, containing the start vertex; a dag, given as an associations list with one association representing an edge; and a list of visited vertices, initially empty. E.g.:

```
val dag=[("a","b"),("a","c"),("a","d"),
         ("b","e"),("c","f"),("d","e"),
         ("e","f"),("e","g")];

> bf(["a"],dag,[]);
val it = ["a", "b", "c", "d", "e", "f", "g"] : string list
```

7. (Haskell) Write a function called `df` which lists all nodes of a dag in a depth-first manner. It has 3 parameters: a list with only one element, containing the start vertex; a dag, given as an associations list with one association representing an edge; and a list of visited vertices, initially empty. E.g.:

```
dag=[("a","b"),("a","c"),("a","d"),("b","e"),("c","f"),
("d","e"),("e","f"),("e","g")]

Main> df ["a"] dag []
["a","b","e","f","g","c","d"]
```