DM580 – EXERCISE SHEET #1

(1) Go to `https://www.haskell.org/ghcup/` and follow the instructions to install `ghc` (select the recommended version).
   (a) Verify that `ghc` is installed correctly by running `ghci` in the terminal (or command line if on Windows). You should see something like this:

```
Loaded package environment from /Users/kaarsgaard/.ghc/[...]
GHCi, version 9.4.8: https://www.haskell.org/ghc/  :? for help
ghci>
```

(2) Which of the following Haskell expressions reduce to the same value? Write down your answers *before* you run them in `ghci`. Did you get the results you expected?
   (a) `2*3*4`
   (b) `2*(3*4)`
   (c) `(2*3)*4`
   (d) `5+6*7`
   (e) `5+(6*7)`
   (f) `(5+6)*7`
   (g) `2*3^4+5`
   (h) `(2*3^4)+5`
   (i) `2*3^(4+5)`
   (j) `sum reverse [7,8,9]`
   (k) `sum (reverse [7,8,9])`

(3) What are the types of the following values? Write down your answers *before* you run them in `ghci`. Did you get the results you expected?
   (a) `[1,2,3]`
   (b) `['1','2','3']`
   (c) `['1','2','3',4]`
   (d) `('1','2','3')`
   (e) `('1','2','3',4)`
   (f) `(head,tail)`
   (g) `[id,reverse]`

(4) For each of the following types, come up with a value which has that type (it doesn't matter which one it is, so long as `ghci` accepts it).
   (a) `[Integer]`
   (b) `[[String]]`
   (c) `Bool -> Bool -> Bool`
   (d) `Integer -> Integer`
   (e) `a -> (a,a)`
   (f) `(a,b) -> (b,b)`
   (g) `(a -> b) -> a -> b`

(5) Write a function `prodpair` that takes a list of integers and returns the product of the first two elements of the list.

(6) Write a function `addone` that takes a list of integers and returns the list where the first element has been incremented by 1.

(7) Write function `lastelem` that returns the last element of a list.
   (a) There are (at least!) two ways of writing this function. Try to come up with both.

(8) Write a function `sumupto` such that `sumupto n` returns the sum of all natural numbers up to and including n, i.e., $0 + 1 + 2 + \cdots + n$.
   (a) As before, there are (at least!) two ways of writing this function. Try to come up with more than one. Do you expect one implementation to be faster than the other? Why or why not?

(9) Write functions `firsthalf` and `secondhalf` which, given a list of length n, returns the list consisting of the first $\frac{n}{2}$ elements respectively the last $\frac{n}{2}$ elements. You decide what these functions do when n is not even, so long as `firsthalf xs ++ secondhalf xs == xs` is satisfied for all lists `xs`.

(10) Argue that your implementation satisfies `firsthalf xs ++ secondhalf xs == xs` for all lists `xs`.

(11) For each of your functions from exercises (5)–(9), write type annotations and ensure (e.g., by running your file with annotated function types in `ghci`) that the Haskell type checker agrees with them.