# **Tutorial 2A: List recursion**

In this week's tutorials we will practice recursion on lists. We will look at recursion on two lists simultaneously, pattern-matching on more than one element, and working with lists of pairs. Finally, we will implement the merge sort algorithm. Along the way, we will introduce **type variables** and simple **type classes**.

#### Lists

Lists are defined inductively, and consist of either the **empty list** [] or a **cons** x:xs of a head and a tail. Your file "2A-List recursion.hs" provides two lists on which to test your functions.

```
ladies = ["Mary","Kitty","Lydia","Elizabeth","Jane"]
gentlemen = ["Charles","Fitzwilliam","George","William"]
```

#### **Exercise 1:**

- a) Complete the function member xs y which returns True if the string y is in the list xs, and False otherwise. In the given patterns, the underscore (\_) is a variable that isn't used. Like any variable, it matches any pattern.
- b) Write a second version named member, this time using boolean "or" (| |) to replace the guards (hint: y is a member of x:xs if it is equal to x or a member of xs).
- c) Complete remove xs y which removes the string y from the list xs. If y is not in the list, return all of xs. (We assume there are no duplicates in xs.) Create your own pattern-matching and guards, as needed.

### Test your functions:

```
*Main> member ladies "Lydia"
True

*Main> member gentlemen "Richard"
False

*Main> remove ladies "Elizabeth"
["Mary","Kitty","Lydia","Jane"]

*Main> remove ladies "Charlotte"
["Mary","Kitty","Lydia","Elizabeth","Jane"]
```

#### **Exercise 2:**

- a) Complete the function members xs ys which returns True if the strings in the list ys also belong to the list xs. Use your function member.
- b) Write a second version named members' using "and" (&&) to replace the guards. Create your own pattern-matching.
- c) Complete the function removeAll xs ys that removes every string in ys from the
  list xs. Use your function removeOne. Create your own pattern-matching. Hint:
  you should not be using guards.

## Test your functions:

```
*Main> members gentlemen ["William", "Charles"]
True
*Main> members gentlemen ["Fitzwilliam", "Elizabeth"]
False
*Main> removeAll ladies ["Lydia", "Charlotte", "Mary"]
["Kitty", "Elizabeth", "Jane"]
```

### **Exercise 3:**

- a) Complete before xs ys that returns True if the string xs comes **strictly** before ys in the dictionary. Create guards for the three cases x < y, x == y, and otherwise.
- b) Give a version before' using boolean operations (||) and (&&) instead of guards.
- c) Complete the function sorted that tests if a list of strings is ordered alphabetically (without duplicates). Create your own pattern-matching, using three cases: empty; one item; and two or more items. Test your code:

```
*Main> before "Elizabeth" "Lydia"
True
*Main> before "George" "George"
False
*Main> sorted ladies
False
*Main> sorted gentlemen
True
```