Assignment 1
COMP 302 Winter 2017
Programming Languages and Paradigms

Prakash Panangaden
McGill University: School of Computer Science

Due Date: 31st January 2017

Please do all the first six questions. Please check the course web page for information on how to format and submit your solution electronically. There are a total of six required questions. There is a seventh question which you can safely ignore; it is intended to stimulate you in other directions than the course material. If you cannot understand question 7, ignore it; this will not affect your comprehension of the class material.

Notes for submission: Your code **must** be formatted using the template file called `assignment1.fs` which should be downloaded from the course web page. **All your code must use the same names and types as I have indicated below. You will get zero otherwise.** Your code must compile or else the grading program will crash. We will give zero if your submission causes the program to crash. Your code must **not** be submitted in Word or any propriety format; it must be plain text so we can test it. The file extension must be .fs; please do **not** submit compiled code. The purpose of this homework is to get a feel for functional programming. **Mutable variables, arrays and refs should not be used**. There are many useful functions in the built-in List library. **You are not allowed to use them for this assignment except for List.map**. In later assignments you can use them freely, but here I want you to get basic practice with writing recursive programs on lists from scratch.

Q1. [15 points]

Given a sequence of real numbers $x_1, x_2, \ldots, x_n$ and another sequence of positive real numbers $w_1, w_2, \ldots, w_n$ called *weights* the *weighted mean* is defined by the following formula

$$\mu = \frac{\sum_{i=1}^{i=n} x_i * w_i}{\sum_{i=1}^{i=n} w_i}.$$

Write an F# program (function) called `w_mean` that takes a list of weights as its first argument and returns a function that takes a list of reals (of the same length as the list of weights) and computes the weighted mean. You can assume that the length of the list of weights is the same as the length of the list of numbers and that all the weights are strictly positive. To deal with error situations use the construct `failwith`. On the web page we have written a little of the code to start you off. The phrase "real number" is, of course, from mathematics. In F# they are implemented as a type called `float`.

You should have auxiliary functions called: (i) `sumlist` which computes the sum of a list of floats, (ii) `pairlists` which takes two lists of floats the same length and produces a list of matched pairs, (iii) `w_mean` which computes the weighted mean as described above.

```
val sumlist : l:float list -> float
val pairlists : 'a list * 'b list -> ('a * 'b) list
val w_mean : weights:float list -> data:float list -> float
```

Q2. [12 points]

Implement in F# a function that tests whether an element is a member of a given list. Here is the type I want.

```
val memberof : 'a * 'a list -> bool when 'a : equality
```

Here are examples of the function in action.

```
> memberof 1 [1;2;3];;
error FS0003: This value is not a function and cannot be applied

> memberof (1,[1;2;3]);;
val it : bool = true
> memberof (1, []);;
val it : bool = false
> memberof (1,[2;3;4]);;
val it : bool = false
```

Implement a function `remove` that takes an element and a list and removes *all copies* of the element from the list. If the element is not in the list the function should return the same list. *Do not use memberof to implement remove.* The type should be as follows:

```
val remove : 'a * 'a list -> 'a list when 'a : equality
> remove (2, [1;3;2;4]);;
val it : int list = [1; 3; 4]
```

Q3. [13 points]

Write an F# function `findMax` that finds the largest value of a list of elements of a comparion type.

```
val findMax : l:'a list -> 'a when 'a : comparison
> let testList = [1;17;3;6;1;8;3;11;6;5;9];;
val testList : int list = [1; 17; 3; 6; 1; 8; 3; 11; 6; 5; 9]
> findMax testList;;
val it : int = 17
```

This is not well defined when the list is empty. We wrote a basic handler for you.

Q4. [20 points]

Write an F# function that implements selection sort and removes duplicates. In this sort the largest element is chosen and put in front and the rest of the list is recursively sorted after the

largest element is removed from the rest of the list. The result will be in descending order. In the example below the duplicates have been removed.

```
val selsort : l:'a list -> 'a list when 'a : comparison
> selsort ([1..2..25]@[1..5..21]);;
val it : int list = [25; 23; 21; 19; 17; 16; 15; 13; 11; 9; 7; 6; 5; 3; 1]
```

Q5. [15 points]

Write a function `common` that takes a pair of lists and forms a new list containing a *unique* copy of each element that occurs in both lists. Here is the type and an example.

```
val common : 'a list * 'a list -> 'a list when 'a : equality

> common ([3;4;5;7;2],[1;3;5;7;9;1]);;
val it : int list = [3; 5; 7]
```

It does not matter if the final list is sorted or not. The input lists are not necessarily sorted.

Q6. [25 points]

The mergesort algorithm is a recursive algorithm for sorting lists which runs in time $O(n \log n)$. The items in the list must have an order relation defined on them, otherwise sorting does not make sense of course.

The idea is as follows: the given list l is split into two equal (if the length of l is odd then one of the "halves" is one item longer than the other) lists l_1 and l_2. These lists are sorted recursively and then the results are merged back to give a single sorted list. Code this in F#. Your algorithm can use $<$ as a comparison operator. Your code *must* have a function `split` that produces a pair of lists, a function `merge` that merges *sorted* lists and a function `mergesort` that implements the overall algorithm.

```
val split : l:'a list -> 'a list * 'a list
val merge : 'a list * 'a list -> 'a list when 'a : comparison
val mergesort : l:'a list -> 'a list when 'a : comparison

> mergesort [15 .. -2 .. 1];;
val it : int list = [1; 3; 5; 7; 9; 11; 13; 15]
> split [15 .. -2 .. 1];;
val it : int list * int list = ([15; 11; 7; 3], [13; 9; 5; 1])
```

Q7. [0 points] This question is for your spiritual growth only. Do not think it will give you extra credit or help you learn the material better. It will however stretch your brain in other directions. Do not attempt it if you have not yet finished the required homework. Do not submit a solution; please talk to me if you have solved it. Do not worry about it if you don't understand the question.

How many comparisons are *required* to find the largest member of a list of n elements? How many are required to find the largest *and* the smallest. Here "required" means that in the worst case you will *have to do* that many comparisons. Can you find the smallest and the largest with fewer than $2n - 3$ comparisons? Note that we are interested in exact counts; not just in $O(\cdot)$ estimates.