COMP 302 Programming Languages and Paradigms
Assignment 2

Prakash Panangaden
McGill University: School of Computer Science

Due Date: 24^{th} **January 2020**

This assignment contains 4 questions; please do them all.

Q1 (a). [12 points] In this question you will write a program that takes two lists of the same length and returns a single list with the items paired together. This will be used in later questions in this assignment; we will need it with floating point items but your code should be polymorphic. The snippet below shows the function name, **which you must use**, and the type returned by the OCaml system. Your code **must** have this type.

```
# let rec pairlists twolists = ...

val pairlists : 'a list * 'b list -> ('a * 'b) list = <fun>
```

We will forbid the use of the `List.combine` library function that does the same thing. Test inputs are two `int` lists of the same length and we will check that you have handled the case of empty input lists. Here is an example showing the right way and the wrong way to use `pairlists`.

```
# let l1 = [1;2;3];;
val l1 : int list = [1; 2; 3]
# let l2 = ['a';'b';'c'];;
val l2 : char list = ['a'; 'b'; 'c']
# let p1 = pairlists (l1,l2);;
val p1 : (int * char) list = [(1, 'a'); (2, 'b'); (3, 'c')]
# let e1 = pairlists l1 l2;;
Characters 9-18:
  let e1 = pairlists l1 l2;;
           ^^^^^^^^^
Error: This function has type 'a list * 'b list -> ('a * 'b) list
       It is applied to too many arguments; maybe you forgot a `;'.
```

Q1 (b). [22 points] If (w_i) is a sequence of *weights*, these are all strictly positive floating point numbers, and (v_i) is a sequence of *values*, which may be positive, zero or negative, we define the weighted mean or average by the formula

$$mean = \frac{\sum_i w_i * v_i}{\sum_i w_i}.$$

In this question you are given two lists of floats as input. The first list is a list of weights and the second is the list of values. Clearly, these lists should be the same length. Implement a function `w_mean` which calculates the weighted mean of the values with the given weights. Here is the declaration and the type that we expect and an example of using it. Note the type and the way that the function is used.

```
val w_mean : float list -> float list -> float = <fun>

  w_mean [1.0;1.5;2.5;0.5;1.5] [10.3;11.7;2.0;5.0;6.5] ;;
- : float = 6.44285714285714217
```

In order to do this you are required to use your `pairlists` function from Q1(a) and the `sumlist` function that we have provided. In this exercise we want you to learn to use the `List.map` function from the library; you are **not** allowed to use `let rec` in this exercise. We guarantee that we will not give you empty lists as test cases nor lists with negative or zero weights.

Q2. [27 points] This question involves two simple exercises in list manipulation. In the first one you are given a pair: the first member of the pair is an item and the second member is a list of items of the same type as the first member. The output is a boolean that says whether the item is in the list or not. The second function takes a pair of an item and a list and returns a list with the item removed. If the item is not in the list then the same list is returned. You can use the first function in implementing the second but we are not requiring you to do so.

Here are the declarations and types.

```
# let rec memberof pair = ...
  val memberof : 'a * 'a list -> bool = <fun>
# let rec remove(item, lst) = ...
  val remove : 'a * 'a list -> 'a list = <fun>
# remove (3, [1;6;3;2;6;1;7;2;3;5]);;
- : int list = [1; 6; 2; 6; 1; 7; 2; 5]
```

The lists may contain duplicates; all copies should be removed. Make sure that you deal with empty lists. Even though your functions are polymorphic the grader cannot handle variant types in the mutation tester. So your test examples should all involve integers only.

Here is an important style remark: **never** write `if foo then true else false`; this should just be `foo`.

Q3. [16 points] In this question we want you to find the largest item in a list of integers. You can assume that the lists are nonempty. The top level declaration is nonrecursive because there is an internal recursive function.

```
# let find_max l = ...
  val find_max : 'a list -> 'a = <fun>
# find_max [1;6;3;2;6;1;7;2;3;5];;
- : int = 7
```

Even though your code is polymorphic and should work on all types for which an order is defined (int, char, string) the grader will get confused if you try to give it a list containing items of dufferent types. Please only test this on integer lists.

Q4. [23 points] In this question we want you to use `find_max` and `remove` to implement a selection sort algorithm. You can assume that the list contains no duplicates.

```
# let rec selsort l =
  val selsort : 'a list -> 'a list = <fun>
# selsort [1;4;2;5;3;9;6;8;7];;
- : int list = [9; 8; 7; 6; 5; 4; 3; 2; 1]
```

You are not allowed to use built-in sorting functions.

Even though your code is polymorphic and should work on all types for which an order is defined (int, char, string) the grader will get confused if you try to give it a list containing items of dufferent types. Please only test this on integer lists.