CIS 120 - Lab 4

Before diving into these exercises, you should follow the instructions on the Lab04 web page (cis.upenn.edu/~cis120/current/hw/lab04) and read account.ml. Explain here why those last tests don’t type check:

Assume the following definitions. Note that the module implementation is left out, as you don’t need those details to be able to use the module.

```ocaml
module type Set = sig
  type 'a set
  val empty : 'a set
  val is_empty : 'a set -> bool
  val member :'a -> 'a set -> bool
  val add : 'a -> 'a set -> 'a set
  val remove : 'a -> 'a set -> 'a set
  val equal : 'a set -> 'a set -> bool
  val elements : 'a set -> 'a list
  val fromList : 'a list -> 'a set
  val setSize : 'a set -> int
end

module Set1 : Set = ...
```
1. Consider the following tests for the Set1 module. Assuming a correct implementation of a Set, which ones pass? Which ones do not compile? Which tests fail? What is the cause of the errors? Write your answers to the right of the code.

```ocaml
;; open Assert
;; open Set1

(* a *)
let test () : bool =
  let set1 = add 1 empty in
  let set2 = add 2 set1 in
  let set3 = add "x" set2 in
  member "x" set3
;; run_test "a" test

(* b *)
let test () : bool =
  let set1 = add 1 empty in
  let set2 = add 2 set1 in
  let x = 1 in
  let set3 = add x set2 in
  member 1 (remove 1 set3)
;; run_test "b" test

(* c *)
let test () : bool =
  let set1 = add 1 empty in
  let set2 = add 2 set1 in
  let x = 1 in
  member x (add "x" set2)
;; run_test "c" test

(* d *)
let test () : bool =
  let set1 = add 1 empty in
  let set2 = add 2 set1 in
  member 2 set1
;; run_test "d" test
```
2. Write a function `union` which computes the union of two sets. Recall that a union of two sets is a set that contains the elements that are in either of the original sets. Part of the problem is to write the correct type for this function.

3. Write a function `setDiff` which, given an `int list` and an `int set`, returns an `int list` containing only those values in the original `int list` that are not in the `int set`. For example, given [1;2;3;4] and {2,3}, this function should return [1;4].
4. Write a function no_evens that, given an int set, returns an int set with all even numbers removed. Recall that a number \( n \) is even if and only if \( n \mod 2 = 0 \).

5. Write a function same_elements that takes two lists (of any type) and returns a bool indicating whether or not the lists have the same elements, ignoring both the order of elements and the appearance of any duplicates.