## Section 3 - Extra practice with higher order functions

This handout was composed by Porter Jones. There are probably plenty of typos/incorrect solutions/etc for you to catch! Please email me with any issues, comments. or feedback at pbjones@cs.washington.edu. All thoughts are welcome :)

Write each of the following SML functions. Use map, filter, or fold when possible to get some practice!

- a) Write a function swap\_pairs\_list that takes a list of pairs and returns a list of pairs with each of the original pairs' values swapped.
- b) Write a function size that takes a list and returns the number of elements in that list.

c) Write a function remove\_all that takes a value and a list and returns a list of the values in the original list not equal to the given value.

d) Write a function contains that takes a value and a list and returns true if the given value is in the list (false otherwise).

e) Write a function intersect that takes two lists as parameters and returns a list that has all of the values contained in both of the given lists. You should use contains in your answer.

## Section 3 - Extra practice solutions

This handout was composed by Porter Jones. There are probably plenty of typos/incorrect solutions/etc for you to catch! Please email me with any issues, comments. or feedback at pbjones@cs.washington.edu. All thoughts are welcome :)

- a) fun swap\_pairs (xs) =
   map (fn (a, b) => (b, a), xs)
  b) fun size xs =
   fold(fn (acc, x) => acc + 1, 0, xs)
  c) fun remove\_all (x, xs) =
   filter(fn y => not(y = x), xs)
  d) fun contains (x, xs) =
   fold(fn (acc, x') => acc orelse x = x', false, xs)
- e) fun intersect (xs, ys) =
   filter(fn x => contains (x, ys), xs)