CSE 341 Section 4 *Common Higher Order Functions Reference Shee*t

map(f, xs) val map = fn : ('a -> 'b) * 'a list -> 'b list

```
fun map(f,xs) =
  case xs of
  [] => []
   | h::t => f h :: map(f,t)
```

Applies the given function f to all elements in the given list xs and returns the resulting list.

```
val double_length = fn x => 2 * size x; (* fn : string -> int *)
val xs = ["hello", "!", "!!!"]; (* string list *)
val result = map(double_length, xs); (* [10,2,6] : int list *)
```

flat_map(f, xs) val flat_map = fn : ('a -> 'b list) * 'a list -> 'b list

Similar to map, but the argument function f returns a list. Then, instead of returning a list of lists ('b list list), “flattens” the list at the end into a 'b list.

```
val repeat_length = fn x => let val len = size x in [len, len] end; (* fn : string -> int list *)
val xs = ["hello", "!", "!!!"]; (* string list *)
val result = flat_map(repeat_length, xs); (* [5,5,1,1,3,3] : int list *)
```

filter(f, xs) val filter = fn : ('a -> bool) * 'a list -> 'a list

```
fun filter(f,xs) =
  case xs of
  [] => []
    | h::t => if f h then h::filter(f,t) else filter(f,t)
```

Applies the given function f to all elements in the given list xs, and only keeps (and returns as a list) the elements that f returned true for.

```
val is_even = fn x => x mod 2 = 0; (* fn : int -> bool *)
val xs = [5, 2, 8]; (* int list *)
val result = filter(is_even, xs); (* [2,8] : int list *)
```

fold(f, acc, xs) val fold = fn : ('a * 'b) -> 'a * 'a * 'b list -> 'a

```
fun fold(f,acc,xs) =
  case xs of
  [] => acc
    | h::t => fold(f, f(acc,h), t)
```

Accumulates an answer by repeatedly applying the given function f to each element in the list, building up to a final result. acc can be thought of as the starting value. In other words, the call to fold(f,acc,[x1,x2,x3,x4]) computes f(f(f(f(acc,x1),x2),x3),x4).

```
val count_greater_than_3 = fn (acc, x) => if x > 3.0 then acc + 1 else acc; (* fn : int * real -> int *)
val xs = [5.0, 2.0, 8.0]; (* real list *)
val result = fold(count_greater_than_3 , 0, xs); (* 2 : int *)
```

Note: size is an SML library function that takes a string and returns the length of it (as an int)