Foundations of Computer Science – Supervision 3 Supplementary Sheet

Heidi Howard

October 30, 2014

Welcome to your third supervision of four in Foundations of Computer Science, we will begin by discussing this weeks problem sheet and answering your questions from the lectures. Later we will move onto looking at a real past exam question, focusing on datatypes.

Exercise 1:

This weeks warmup exercise are as follows:

```
(* given a binary search tree, min returns
        the minimum element in the tree *)

val min = fn: 'a tree -> 'a

(* given a function and a list, map applies
        the function to each element in the list
        and returns the resulting list *)

val map = fn: ('a -> 'b) -> 'a list -> 'b list

(* given a predicate and a list, filter applies
        the predicate to each element in the list and removes
        elements if the predicate returns false *)

val filter = fn: ('a -> bool) -> 'a list -> 'a list

(* given an element and a list, remove_all removes
        all occurences of the element from the list *)

val remove_all = fn: 'a -> 'a list -> 'a list

(* given a sequence, head returns the first element *)

val head = fn: 'a seq -> 'a

(* given a sequence, tail returns the
        sequence without the first element *)
```

```
val tail = fn: 'a seq -> 'a

(* given an element and a sequence,
        add the element to the front of the sequence *)

val cons = fn: 'a -> 'a seq -> 'a seq
```

Exercise 2:

Write an implementation of take and drop for lazy lists.

[adapted from the definitons on pg 38, sl 401 in the lecture notes]

Exercise 3:

We often talk about how the options datatype can be used to replace exceptions in SML. Views differ on whether options or exceptions should be used for error handling. Implement the following functions to convert functions which use exception to onces which use options and vise versa.

```
val to_option = fn: ('a -> 'b) -> 'a -> 'b option
val to_exception = fn: ('a -> 'b option) -> 'a -> 'b
```