# Foundations of Computer Science – Supervision 2 Supplementary Sheet

Heidi Howard

October 24, 2014

Welcome to your second supervision 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:**

Write out the following 8 functions as quickly as possible (as we did last week)

```
(* member takes a list and a element and returns true
        if the element is a member of the list and false otherwise. *)
val member = fn: ''a * ''a list -> bool

(* inter takes two list and returns the intersection of the two list,
        which means the resulting list has all elements that present in
        both of the lists but this doesn't introduce duplicates *)
val inter = fn: 'a list * 'a list -> 'a list

(* union takes two list and returns the union of the two list,
        which means the resulting list has all elements that present
        in at least one of the given lists but this doesn't
        introduce duplicates *)
val union = fn: 'a list * 'a list -> 'a list

(* flatten takes a list of lists and returns a single list with
        all the same elements.
        e.g: flatten [[1,2],[3,4],[],[7.8]] returns [1,2,3,4,7,8] *)
val flatten = fn: 'a list list -> 'a list

(* given two list, add_lists returns a new list where the
        elements of the two lists have been added together pairwise *)
(* if lists are different length, then the result
        is equivient to padding the shorter list with zeros *)
(* e.g. add_lists ([1,2,3,4],[1,2,3]) will return [2,4,6,4] *)
val add_lists = fn: int list * int list -> int list

(* Here is the ML datatype for trees *)
```

```
datatype 'a tree = Lf
                 | Br of 'a * 'a tree * 'a tree

(* root takes a tree and returns the value of the root node *)
val root = fn: 'a tree -> 'a

(* sum takes a integer tree and returns the
        sum of the all the integers in the tree *)
val sum = fn: int tree -> int

(* cons takes an element and a tree and
        returns a tree with that element as the new root *)
val cons = fn: 'a * 'a tree -> 'a tree
```

**Exercise 2:**
Complete the tripos question from 2002 P1 Q1.

**Exercise 3:**
Complete the tripos question from 2002 P1 Q5 (parts (a) and (c) only).

**Exercise 4:**
Research and explain the algorithm Counting Sort. What is its time and space complexity? Explain why it does not contradict slide 502. What time/space complexity might counting sort have if implemented in ML using lists to store the counts?