# Foundations of Computer Science – Problem Sheet 2

Heidi Howard

October 21, 2014

This supervision is all about lists, sorting and datatypes. You should come to this supervision prepared to explain the making change and quicksort examples for the course notes.

**Exercise 1:**
Write a function which takes two list and return true if the first list is present anywhere within the second and false otherwise. For example `sublist ([1,2],[0,1,2,3])` returns true but `sublist ([3,5],[5,4,3,2,1])` returns false.

**Exercise 2:**
Write a function to test if a list is a palindrome. You may make use of functions in the lecture course so far.

A palindrome is a word (or here, for the sake of exercising, a list of elements) that is the same when it's read in both ways (i.e., from left to right or from right to left). Since we're using lists to represent arbitrary words, any list with a single element is a palindrome, and an empty list is also a palindrome.

**Exercise 3:**
We know nothing about the functions f and g other than their polymorphic types:

```
> val f = fn: 'a * 'b -> 'b * 'a
> val g = fn: 'a -> 'a list
```

Suppose that `f(1, true)` and `g 0` are evaluated and return their results. State, with reasons, what you think the resulting values will be.

***[taken from exercise 4.6 from the course notes]***

**Exercise 4:**
Implement selection sort using ML.

The selection sort consists of looking at the elements to be sorted, identifying and removing a minimal element, which is placed at the head of the result. The tail is obtained by recursively sorting the remaining elements. If you are not familiar with the algorithm, look it up online.

Comment on the space and time complexity of your implementation.

***[taken from exercise 5.2 from the course notes]***

**Exercise 5:**
Implement bubble sort using ML.

The bubble sort consists of looking at adjacent pairs of elements, exchanging them if they are out of order and repeating this process until no more exchanges are possible. If you are not familiar with the algorithm, look it up online.

Comment on the space and time complexity of your implementation.

***[taken from exercise 5.4 from the course notes]***

**Exercise 6:**

Write a datatype for representing the months of the year. Write a function called `to_day` to convert from a month and day to the day number in the year[1] and a function called `to_date` to convert back from the day number in the year to a month and day.

For example, `to_day (Feb,2)` returns 33 and `to_day (Dec,31)` return 365 and `to_date 12` returns `(Jan,12)` and `to_date 284` return `(Oct,21)`. If you are short of time, then assume all months have exactly 30 days and handle only the first 4 months.

***[adapted from exercise 6.1 from the course notes]***

**Exercise 7:**

The following is a datatype for representing lists.

```
datatype 'a mylist = Nil | Cons of 'a * 'a mylist
```

This definition was taken from page 68 (under slide 609) in the course notes.

(a) Write new `hd`, `tl` and `null` functions (called `myhd`, `mytl` and `mynull`) for this new representation of lists. Your `myhd` or `mytl` functions should handle empty list the same way as ML's `hd` and `tl` functions do.

(b) Write new append and reverse functions (called myappend and myrev) for this new representation of lists.

(c) Write the functions `to_mylist` and `to_list` which convert between this new representation of lists and the built-in representation of lists.

**Exercise 8:**

The ML data type `Boolean`, defined below, is to be used to represent boolean expressions.

```
datatype Boolean =
    Var of string
  | Not of Boolean
  | And of Boolean * Boolean
  | Or of Boolean * Boolean
```

The constructor `Var` is used to represent named boolean variables, and `Not`, `And` and `Or` are used to represent the corresponding boolean operators.

(a) Define a function that will return a list of the distinct names used in a given boolean expression.

(b) A context is represented by a list of strings corresponding to the boolean variables that are set to true. All other variables are deemed to be set to false. Define a function that will evaluate a given boolean expression in a given context.

(c) Incorporate your two functions into a program that will determine whether a given boolean expression is true for all possible settings of its variables.

[1] if you are unsure what a day number is see http://www.epochconverter.com/epoch/daynumbers.php

*[taken from tripos question 1996 Paper 1 Question 2]*

**Exercise 9:**

Exercise 8 creates a datatype for boolean algebra and goes on to evaluate expressions in boolean algebra. We will now look at this in the context of arithmetic expressions instead.

(a) Give an ML datatype for representing arithmetic expression: an expression in our case is either a integer, the additions of two expressions, the multiplication of two expressions or the negation of an expression.

(b) Write a function called `eval` which transform an arithmetic expression to an integer e.g.

```
> eval (Add (Int 4, Int 6));
val it = 10: int
> eval (Multi (Add (Int 1, Int 2), Neg (Int 3)));
val it = ~9: int
```

(c) Extend your datatype and eval definition to allow for variables. A context is represented by a list of string integer tuples. An example of the results is:

```
> val context = [("x",7),("y",8)];
val context = [("x", 7), ("y", 8)]: (string * int) list

> eval (context, Add (Var "x", Var "y"));
val it = 15: int
```

Note that context is an example of an association list, which is a simple dictionary representation detailed on slide 701. If the expression contains any string which are not mapped to integers, your function should raise an exception.

*[adapted from exercise 6.4 & 6.5 from the course notes]*