Programming Language Concepts, CS2104 (8th October 2007)
Tutorial 6

Exercise 1. (ADT) Design an ADT for Set in Oz. Consider carefully the typical operations, such as membership test, union, intersection etc. Provide first an interface informally in terms of types. After that, implement it using say a list (perhaps sorted if you like

Exercise 2. (Haskell Type Class) Given a list of elements, one way of designing a membership test operation is to use the following:

```
isMember :: A -> [A] -> Boolean
```

Explain why this type structure is wrong? (Hint : equality of function is undecidable). Hence, suggest a more appropriate type for the above function together with its code. How would you make a (List A) type to be an instance of the Eq class? Also, how would you make a (Set A) type to be an instance of the Ord class?

Exercise 3. (Tail-Recursion) Consider the following linear-recursive function:

```
fun {Double Ls}
   case Ls of nil then nil
            [] H|T then (2*H)|{Double T} end
end
```

The following are three attempts to implement the above function using tail-recursion.

```
fun {Double1 Ls Acc}
   case Ls of nil then Acc
            [] H|T then {Double1 T (Acc|2*H)} end
end
fun {Double2 Ls Acc}
   case Ls of nil then Acc
            [] H|T then {Double2 T {Append Acc [2*H]}} end
end
fun {Double3 Ls F}
   case Ls of
      nil then {F nil}
     [] H|T then {Double3 T (fun {$ R} {F (2*H|R)} end)} end
end
```

One of the approach is wrong, while the other two approaches have poor computational characteristic. Discuss these solutions and indicate the format of the initial call. Test them on Mozart