1 Appendix

```
signature Sequence =
sig
    type 'a t
    type 'a seq = 'a t

    exception Range of string
                                                        (* span *)
    val empty : unit -> 'a seq                          (* O(1) *)
    val tabulate : (int -> 'a) -> int -> 'a seq          (* O(1) *)
    val nth : 'a seq -> int -> 'a                       (* O(1) *)
    val length : 'a seq -> int                          (* O(1) *)

    val fromList : 'a list -> 'a seq                    (* O(n) *)
    val toList : 'a seq -> 'a list                      (* O(n) *)

    val map : ('a -> 'b) -> 'a seq -> 'b seq            (* O(1) *)
    val reduce : ('a * 'a -> 'a) -> 'a -> 'a seq -> 'a  (* O(log n) *)
    val reduce1 : ('a * 'a -> 'a) -> 'a seq -> 'a       (* O(log n) *)
    val filter : ('a -> bool) -> 'a seq -> 'a seq       (* O(log n) *)
    val rev : 'a seq -> 'a seq                          (* O(1) *)
    val enum : 'a seq -> (int * 'a) seq                 (* O(1) *)

    val subseq : 'a seq -> int * int -> 'a seq          (* O(1) *)
    val take : 'a seq -> int -> 'a seq                  (* O(1) *)
    val drop : 'a seq -> int -> 'a seq                  (* O(1) *)
end
```

```
signature Stream =
sig
    type 'a stream
    datatype 'a front = Empty | Cons of 'a * 'a stream

    val delay : (unit -> 'a front) -> 'a stream
    val expose : 'a stream -> 'a front

    val empty : 'a stream
    val cons : 'a * 'a stream -> 'a stream
    val fromList : 'a list -> 'a stream
    val tabulate : (int -> 'a) -> 'a stream

    val null : 'a stream -> bool
    val hd : 'a stream -> 'a
    val take : 'a stream * int -> 'a list
    val toList : 'a stream -> 'a list

    val tl : 'a stream -> 'a stream
    val drop : 'a stream * int -> 'a stream
    val append : 'a stream * 'a stream -> 'a stream

    val map : ('a -> 'b) -> 'a stream -> 'b stream
    val filter : ('a -> bool) -> 'a stream -> 'a stream
    val zip : 'a stream * 'b stream -> ('a * 'b) stream
end
```