1.1 OCaml code examples

1.1.1 Calculate the average

Listing 1: dictionary

```ocaml
(* calculate the average of a list of integers *)
let grades = [80; 90; 70; 60];;
let rec fold f l acc =
  match l with
  | [] -> acc
  | h :: t -> f h (fold f t acc)
  |;
let sum l = fold (fun x y -> x + y) l 0;;
let s = sum grades;;
print_int s;;
print_string "\n";;

let avg l =
  let s = sum l in
  let rec length l =
    match l with
    | [] -> 0
    | h :: t -> 1 + length t
    in s / (length l)
  in
    print_int v;;
print_string "\n";;
```

1.1.2 Insertion Sort

Listing 2: insertion sort

```ocaml
let rec sort = function
  | [] -> []
  | x :: l -> insert x (sort l)
and insert elem = function
  | [] -> [elem]
```
Listing 3: insertion sort 2

```ocaml
let rec sort lst = match lst with |
| [] -> []
| x :: l -> insert x (sort l);

let rec insert elem lst = match lst with |
| [] -> [elem]
| x :: l -> if elem < x then elem :: x :: l
else x :: insert elem l;;
```

1.1.3 List of functions

Listing 4: apply list of functions to a list

```ocaml
(* in this example, we will apply a list of functions to a list
and return the result as a list of list *)
let list= [2;3;6;9];;
let double x = x * 2;;
let halve x = x / 2;;
let self x = x;;
let square x = x * x;;
let f list = [double; halve; self; square];;
let rec map map f l l l = match f l with |
| [] -> []
| h1 :: t1 -> (let rec map f l=|
| [] -> []
| h :: t -> f h :: map f t
in map h1 l1
)::map_map t1 l1
in
map_map f list l;;
```

Listing 5: Result

```
Result:
[[[4; 6; 12; 18]; [1; 1; 3; 4]; [2; 3; 6; 9]; [4; 9; 36; 81]]
```

In the example in Listing 4, if we add following two functions to the function list.

Listing 6: more functions

```
let is_even x = if x mod 2 = 0 then true else false;;
```
let to_str x = string_of_int x;;
let flist = [double; halve; self; square; is_even; to_str];;

Does it work? Why?

1.1.4 Explode: String to list

Listing 7: Result

let explode s =
  let rec expl i l =
    if i < 0 then l else
    expl (i - 1) (s.[i] :: l) in
  expl (String.length s - 1) [];;

1.1.5 Implode: Character list to string

Listing 8: Result

let implode l =
  let result = String.create (List.length l) in
  let rec imp i = function
    | [] -> result
    | c :: l -> result.[i] <- c; imp (i + 1) l in
  imp 0 l;;

1.1.6 Read a File

Listing 9: Read a file

(* Read lines from a text file. Use regular expression to replace the first letter of each line with A *)

let change name =
  Str.replace_first (Str.regexp "^[A-Z]"") "ABC" name
;;

let read_file file_name =
  let in_file = open_in file_name in
  try
    while true do
      let line = input_line in_file in
      print_endline (change line)
    done
  with End_of_file ->
    close_in in_file
;;
19 20 21
\texttt{read\_file "names.txt";;}

References

[OCaml from the very beginning]  
JOHN WHITTINGTON \textit{Coherent Press}