- Introduction and overview
- Basic types, definitions and functions
- Basic data structures
- More advanced data structures
- Higher order functions
- **Exceptions, input/output and imperative constructs**

Table of Contents

Imperative features in OCaml

Getting and handling your Exceptions
Week 5 Echéance le déc 12, 2016 at 23:30 UTC

Getting information in and out
Week 5 Echéance le déc 12, 2016 at 23:30 UTC

Sequences and iterations
Week 5 Echéance le déc 12, 2016 at 23:30 UTC

Mutable arrays
Week 5 Echéance le déc 12, 2016 at 23:30 UTC

Mutable record fields
Week 5 Echéance le déc 12, 2016 at 23:30 UTC

Variables, aka References
Week 5 Echéance le déc 12, 2016 at 23:30 UTC

- Modules and data abstraction
- Project

PRINTING LISTS (200/200 points)

1. Write a function `print_int_list : int list -> unit` that takes a list of integers as input, and prints all the elements of the list, each on its own line.
2. Write a function `print_every_other : int -> int list -> unit` that takes a value k and a list of integers as input, and prints the elements of the list that are in positions multiple of k, each on its own line. Note: the first element of a list is at the position 0, not 1.
3. Write a function `print_list : ('a -> unit) -> 'a list -> unit` that takes a printer of values of some type `'a` and a list of such values as input, and prints all the elements of the list, each on its own line.

YOUR OCAML ENVIRONMENT

```
let rec print_int_list l = match l with
  | [] -> ()
  | hd::tl -> print_int hd; print_string "\n"; print_int_list tl
;;

let print_every_other k l =
  let liste = List.fold_left
      (fun (list, pos) element -> if (mod) pos k = 0 then ((list @ [element]), pos + 1)
        else (list, pos + 1))
      ([], 0)
      l
  in match liste with
    (l,_) -> print_int_list l
;;

let rec print_list print l = match l with
  | [] -> ()
  | hd::tl -> print hd; print_string "\n"; print_list print tl
;;

```

Evaluate >

Switch >>

Typecheck

Reset Templ

Full-screen

Check & Sa

Exercise complete (click for details) 200 pts

v Exercise 1: `print_int_list` Completed, 50 pts

Found `print_int_list` with compatible type.

Computing `print_int_list []`

Expected output 5 pts

Computing `print_int_list [-5]`

Expected output 5 pts

```
-5
```

Computing `print_int_list [4]`

Expected output 5 pts

```
4
```

Computing `print_int_list [3]`

Expected output 5 pts

```
3
```

Computing `print_int_list [1; -3]`

Expected output 5 pts

```
1
-3
```

Computing `print_int_list [2; -4; -5; -3; 1; 4; 3; 4; -3]`

Expected output 5 pts

```
2
-4
-5
-3
1
4
```

Computing `print_int_list [2; 0; -3; 2]`
Expected output 5 pts
```
2
0
-3
2
```
Computing `print_int_list []`
Expected output 5 pts
Computing `print_int_list []`
Expected output 5 pts
Computing `print_int_list [-3; -4; -2; 2; 1]`
Expected output 5 pts
```
-3
-4
-2
2
1
```

Exercise 2: `print_every_other` — Completed, 50 pts

Found `print_every_other` with compatible type.
Computing `print_every_other 5 [2; 7; -8; -10; -1]`
Expected output 5 pts
```
2
```
Computing `print_every_other 1 [7; -3]`
Expected output 5 pts
```
7
-3
```
Computing `print_every_other 2 [1; -3; -7; 3; -7; -4]`
Expected output 5 pts
```
1
-7
-7
```
Computing `print_every_other 4 [-9; 6; -3; -9; -7; 8; 1]`
Expected output 5 pts
```
-9
-7
```
Computing `print_every_other 3 [-1; -5; -2; 5; 7; -6; -4; -5]`
Expected output 5 pts
```
-1
5
-4
```
Computing `print_every_other 3 [-5; -6; 5; -10; -8; -3; -3; -9; 6]`
Expected output 5 pts
```
-5
-10
-3
```
Computing `print_every_other 5 [-1]`
Expected output 5 pts
```
-1
```
Computing `print_every_other 5 []`
Expected output 5 pts
Computing `print_every_other 1 [6]`
Expected output 5 pts
```
6
```
Computing `print_every_other 1 [-9; 5; -4; 1; -4; 7; 6; -3]`
Expected output 5 pts
```
-9
5
-4
1
-4
7
6
-3
```

Exercise 3: `print_list` — Completed, 100 pts

testing with integers
Found `print_list` with compatible type.
Computing `print_list print_int [4; 5; 2; 3; 3; 5]`
Expected output 5 pts
```
4
5
2
3
3
5
```

```
  [5]
  [5]
  [4]
  [1]
  [1]
```

Computing `print_list print_int [3; 4; 2; 3]`

Expected output 5 pts

```
  3
  4
  2
  3
```

Computing `print_list (Printf.printf "[%d]") [4; 5; 3; 2; 1]`

Expected output 5 pts

```
  [4]
  [5]
  [3]
  [2]
  [1]
```

Computing `print_list (Printf.printf "[%d]") [4; 1; 3]`

Expected output 5 pts

```
  [4]
  [1]
  [3]
```

Computing `print_list print_int []`

Expected output 5 pts

Computing `print_list (Printf.printf "[%d]") [1]`

Expected output 5 pts

```
  [1]
```

Computing `print_list print_int []`

Expected output 5 pts

Computing `print_list (Printf.printf "[%d]") []`

Expected output 5 pts

Computing `print_list print_int [4; 3]`

Expected output 5 pts

```
  4
  3
```

testing with booleans

Found `print_list` with compatible type.

Computing

```
  print_list
    (Printf.printf "%b")
    [false; false; true; false; false; true; false; false; true; true]
```

Expected output 5 pts

```
  false
  false
  true
  false
  false
  true
  false
  false
  true
  true
```

Computing

```
  print_list
    (function true -> print_string "YES" | false -> print_string "NO")
    [false; false; true; false; true; false]
```

Expected output 5 pts

```
  NO
  NO
  YES
  NO
  YES
  NO
```

Computing `print_list (Printf.printf "%b") [true; true; false]`

Expected output 5 pts

```
  true
  true
  false
```

Computing

```
  print_list
    (function true -> print_string "YES" | false -> print_string "NO")
    [false; false; false; true; false; false; true; false; true]
```

Expected output 5 pts

```
  NO
  NO
  NO
  YES
  NO
```

Computing
```
print_list
  (function true -> print_string "YES" | false -> print_string "NO")
  [false; false; true; false; true; false; true; true]
```
Expected output 5 pts
```
NO
NO
YES
NO
YES
NO
YES
YES
```
Computing
```
print_list
  (Printf.printf "%b")
  [true; true; true; false; true; true; false; false; false]
```
Expected output 5 pts
```
true
true
true
false
true
true
false
false
false
```
Computing `print_list (Printf.printf "%b") [false; true; true; true]`

Expected output 5 pts
```
false
true
true
true
```
Computing
```
print_list
  (function true -> print_string "YES" | false -> print_string "NO")
  [true; false; false; false; false; false; false; true]
```
Expected output 5 pts
```
YES
NO
NO
NO
NO
NO
NO
YES
```
Computing
```
print_list
  (Printf.printf "%b")
  [false; true; true; true; true; true; true; false]
```
Expected output 5 pts
```
false
true
true
true
true
true
true
false
```
Computing
```
print_list
  (function true -> print_string "YES" | false -> print_string "NO")
  [true; false; true; false; false; false; true; true]
```
Expected output 5 pts
```
YES
NO
YES
NO
NO
NO
YES
YES
```