- 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 WITH LOOPS (105/105 points)

1. Using the `for` loop construct, write an `output_multiples : int -> int -> int -> unit` function that prints all the multiples of `x` in the integer interval `n ... m`, separated by commas (`','`).
2. Define a higher order function `display_sign_until_zero: (int -> int) -> int -> unit` that takes a function `f`, an integer `m` and applies `f` from `0` to `m` using a `for` loop. The function will print `"negative"` if the result of `f` if strictly negative and `"positive"` if strictly positive. Each print should appear on a new line.
   Your function has to stop displaying the signs as soon as `f` returns `0`. In this case, it must print a last `"zero"`.
   To implement this, you will define your own exception, `raise` it from inside the loop to break it, and catch it outside of the loop so that the function returns a successful `()`.
   You cannot use a predefined exception.

THE GIVEN PRELUDE

```
let is_multiple i x = i mod x = 0
```

YOUR OCAML ENVIRONMENT

```
let output_multiples x n m =
  for i = n to m do
    if (mod) i x = 0 then
      begin
        print_int i ;
        print_string ","
      end
  done
;;

exception Zero;;

let display_sign_until_zero f m =
  try
    for i = 0 to m do
      if f i = 0 then
        raise Zero
      else
      if f i < 0 then
        begin
          print_string "negative";
          print_string "\n"
        end
      else
        begin
          print_string "positive";
          print_string "\n"
        end
    done;
  with
    Zero -> print_string "zero"
;;

```

Evaluate >

Switch >>

Typechecl

Reset Templ

Full-screen |

Check & Sa

```
Exercise complete (click for details)                         105 pts
v Exercise 1: output_multiples                      Completed, 55 pts
 Found a toplevel definition for output_multiples .
 You used a for loop, bravo!!                                   5 pts
 Now I will check that it behaves correctly
 Found output_multiples with compatible type.
 Computing output_multiples 5 4 38
 Expected output                                                5 pts
   5,10,15,20,25,30,35,
 Computing output_multiples 6 4 37
 Expected output                                                5 pts
   6,12,18,24,30,36,
 Computing output_multiples 2 3 37
 Expected output                                                5 pts
   4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,
 Computing output_multiples 6 3 38
 Expected output                                                5 pts
   6,12,18,24,30,36,
Computing output_multiples 2 6 32
Expected output                                              5 pts
  6,8,10,12,14,16,18,20,22,24,26,28,30,32,
Computing output_multiples 5 6 31
Expected output                                              5 pts
  10,15,20,25,30,
Computing output_multiples 6 3 32
Expected output                                              5 pts
  6,12,18,24,30,
Computing output_multiples 6 5 32
Expected output                                              5 pts
  6,12,18,24,30,
Computing output_multiples 2 5 35
Expected output                                              5 pts
  6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,
v Exercise 2: display_sign_until_zero            Completed, 50 pts
Found a toplevel definition for display_sign_until_zero .
You used all the required syntactic constructs.
Now I will check that your function behaves correctly
Found display_sign_until_zero with compatible type.
Computing display_sign_until_zero (fun i -> i - 6) 6
Expected output                                              5 pts
  negative
  negative
  negative
  negative
  negative
  negative
  zero
Computing display_sign_until_zero (fun i -> (i mod 2) * 2 - 1) 9
Expected output                                              5 pts
  negative
  positive
  negative
  positive
  negative
  positive
  negative
  positive
  negative
  positive
Computing display_sign_until_zero (fun i -> i - 3) 8
Expected output                                              5 pts
  negative
  negative
  negative
  zero
Computing display_sign_until_zero (fun i -> 8 - i) 6
Expected output                                              5 pts
  positive
  positive
  positive
  positive
  positive
  positive
  positive
Computing display_sign_until_zero (fun i -> 4 - i) 6
Expected output                                              5 pts
  positive
  positive
  positive
  positive
  zero
Computing display_sign_until_zero (fun i -> i - 3) 9
Expected output                                              5 pts
  negative
  negative
  negative
  zero
Computing display_sign_until_zero (fun i -> (i mod 2) * 2 - 1) 5
Expected output                                              5 pts
  negative
  positive
  negative
```

Computing display_sign_until_zero (fun i -> 4 - i) 5

Expected output 5 pts

```
positive
positive
positive
positive
zero
```

Computing `display_sign_until_zero (fun i -> 8 - i) 5`

Expected output 5 pts

```
positive
positive
positive
positive
positive
positive
```

Computing `display_sign_until_zero (fun i -> i - 6) 9`

Expected output 5 pts

```
negative
negative
negative
negative
negative
negative
zero
```