



- 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 LTC

Getting information in and out

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

T.

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

SIMPLE USES OF REFERENCES (50/50 points)

- 1. Define swap: 'a ref -> 'a ref -> unit that swaps the contents of two references.
- 2. Define update: 'a ref -> ('a -> 'a) -> 'a that calls a function on the contents of a reference, updates it with the result, and returns the old value.

 For instance let r = ref 6 in update r (function x -> x + 1) should return 6 and set the reference to 7
- 3. Define move: 'a list ref -> 'a list ref -> unit, that removes the top argument from the first list and puts it on top of the second list. If the first list is empty, it should raise Empty.
- 4. A common pattern is to use a reference to perform a computation in an imperative way, but to keep it in a local definition, completely invisible from outside the function implementation.

Define reverse: 'a list -> 'a list], that has a type and an observable behaviour that look like the ones of a purely functional function, buf that use a reference internally to perform the computation. It takes a list, and returns a copy of the list whose elements are in reverse order.

The only functions you can call, except from locally defined functions, are (!), (:=), ref, and move that you just defined. And you are not allowed to use pattern matching.

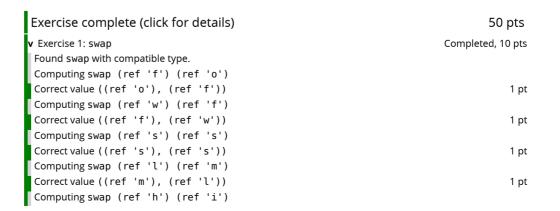
THE GIVEN PRELUDE

```
exception Empty ;;
```

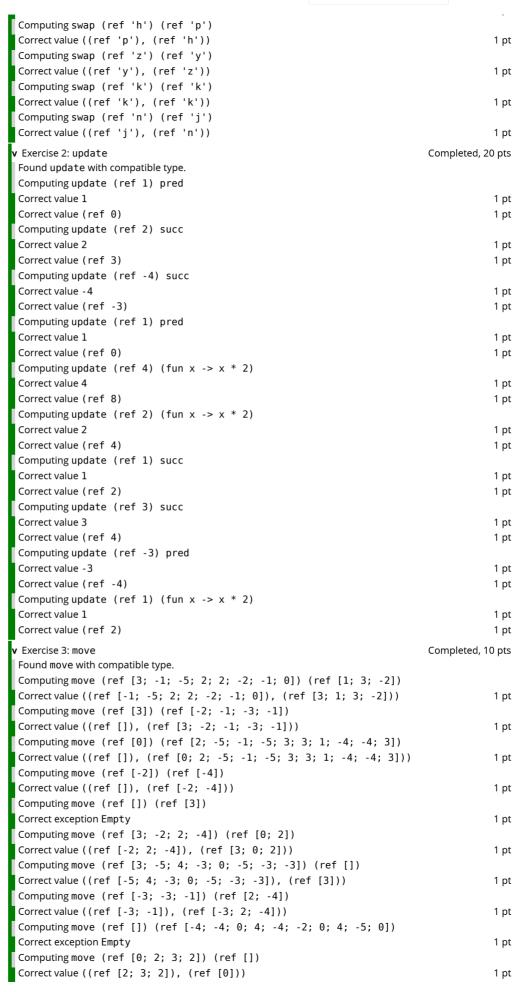
YOUR OCAML ENVIRONMENT

```
let swap ra rb =
let rc = ref !rb in
                                                                                                                                                                   Evaluate >
          rb := !ra;
ra := !rc
                                                                                                                                                                    Switch >>
      let update r f =
  let res = ref !r in
          let res = :
r := f !r;
10
11
                                                                                                                                                                     Typechecl
      14
15
                                                                                                                                                                Reset Templ
18
          et reverse l =
let rev = ref [] in
let liste = ref l in
let doing =
21
                                                                                                                                                                 Full-screen I
23
24
25
             try
while true do
26
27
          done
with _ -> ()
in doing;
28
29
30
31
32
         !rev
                                                                                                                                                                  Check & Sa
```

- Modules and data abstraction
- Project















Tourid Tever 36 with compatible type.	
Computing reverse ['i'; 'n'; 'c']	
Correct value ['c'; 'n'; 'i']	1 pt
Computing reverse ['e'; 'u'; 'q'; 'i'; 'k'; 'r']	
Correct value ['r'; 'k'; 'i'; 'q'; 'u'; 'e']	1 pt
Computing reverse []	
Correct value []	1 pt
Computing reverse ['a'; 'w'; 'p'; 'z'; 'y'; 'a'; 'c'; 'v'; 'f']	
Correct value ['f'; 'v'; 'c'; 'a'; 'y'; 'z'; 'p'; 'w'; 'a']	1 pt
Computing reverse ['c'; 't']	
Correct value ['t'; 'c']	1 pt
Computing reverse ['b'; 'n'; 'q']	
Correct value ['q'; 'n'; 'b']	1 pt
Computing reverse ['q'; 'g'; 'e'; 'r'; 'b'; 'g'; 'r'; 'w'; 'w']	
Correct value ['w'; 'w'; 'r'; 'g'; 'b'; 'r'; 'e'; 'g'; 'q']	1 pt
Computing reverse ['d'; 'q'; 'h'; 'v'; 'c']	
Correct value ['c'; 'v'; 'h'; 'q'; 'd']	1 pt
Computing reverse []	
Correct value []	1 pt
Computing reverse ['v'; 'y'; 'n'; 'w'; 'g'; 'l'; 'k'; 'j'; 'u'; 'o']	
Correct value ['o'; 'u'; 'j'; 'k'; 'l'; 'g'; 'w'; 'n'; 'y'; 'v']	1 pt

A propos

Aide

Contact

Conditions générales d'utilisation

Charte utilisateurs

Politique de confidentialité

Mentions légales







