



- ▶ Introduction and overview
- ▶ Basic types, definitions and functions
- ▶ Basic data structures

▶ More advanced data structures

Table of Contents

Tagged values

Week 3 Échéance le déc 12, 2016 at 23:30 UTC

Recursive types

Week 3 Échéance le déc 12, 2016 at 23:30 UTC

Tree-like values

Week 3 Échéance le déc 12, 2016 at 23:30 UTC

Case study: a story teller

Week 3 Échéance le déc 12, 2016 at 23:30 UTC

Polymorphic algebraic datatypes

Week 3 Échéance le déc 12, 2016 at 23:30 UTC

Advanced topics

Week 3 Échéance le déc 12, 2016 at 23:30 UTC

- ▶ Higher order functions
- ▶ Exceptions, input/output and imperative constructs
- ▶ Modules and data abstraction

THE OPTION TYPE (30/30 points)

Optional values are commonly used in OCaml in the return type of partial functions, i.e. functions that may fail on some input. The following questions illustrate such situations.

In the `Pervasives` module which is loaded automatically, there is a type `option` with two constructors:

`Some (e)` has type `'t option` if `e` has type `'t` and represents the presence of some value `e` of type `'t`.

`None` has type `'t option` and represents the absence of some value of type `'t`.

1. Write a function `find : string array -> string -> int option` such that `find a w = Some idx` if `a.(idx) = w` and `find a w = None` if there is no such index.
2. Sometimes, when a value of type `t` is missing, a default value should be used. Write a function `default_int : int option -> int` such that: `default_int None = 0` and `default_int (Some x) = x`.
3. Write a function `merge : int option -> int option -> int option` such that:
 - `merge None None = None`
 - `merge (Some x) None = merge None (Some x) = Some x`
 - `merge (Some x) (Some y) = Some (x + y)`

YOUR OCAML ENVIRONMENT

```
1 let find a w =
2   let rec find_rec a w = function
3     | idx -> if idx = Array.length a then None else
4             if a.(idx) = w then Some idx else find_rec a w (idx + 1)
5   in
6     find_rec a w 0
7 ;;
8
9 let default_int = function
10  | None -> 0
11  | Some x -> x
12 ;;
13
14 let merge a b = match a,b with
15  | (None, None) -> None
16  | (None, Some x) -> Some x
17  | (Some x, None) -> Some x
18  | (Some x, Some y) -> Some (x + y)
19 ;;
20
```

[Evaluate >](#)[Switch >>](#)[Typechecked](#)[Reset Templ](#)[Full-screen |](#)[Check & Sa](#)

Exercise complete (click for details)

30 pts

Completed, 10 pts

- ▼ Exercise 1: find
Found find with compatible type.
Computing find [||] ""
Correct value None 1 pt
Computing find [|"bowl"; "kilt"; "elephant"; "gorilla"; "fig"|] "bowl"
Correct value (Some 0) 1 pt
Computing find [|"diddy"; "heart"; "isle"; "clown"|] "force"
Correct value None 1 pt
Computing find [|"alpha"; "clown"; "fig"; "elephant"|] "clown"
Correct value (Some 1) 1 pt
Computing find [|"diddy"; "isle"; "heart"|] "heart"
Correct value (Some 2) 1 pt
Computing find [|"kilt"; "gorilla"|] "kilt"
Correct value (Some 0) 1 pt

Correct value (Some 0)	1 pt
Computing find ["gorilla"; "diddy"] "gorilla"	
Correct value (Some 0)	1 pt
Computing find ["fig"; "clown"] "clown"	
Correct value (Some 1)	1 pt
✓ Exercise 2: default_int	Completed, 10 pts
Found default_int with compatible type.	
Computing default_int None	
Correct value 0	1 pt
Computing default_int (Some 10)	
Correct value 10	1 pt
Computing default_int (Some 3)	
Correct value 3	1 pt
Computing default_int (Some 7)	
Correct value 7	1 pt
Computing default_int None	
Correct value 0	1 pt
Computing default_int (Some 9)	
Correct value 9	1 pt
Computing default_int (Some 3)	
Correct value 3	1 pt
Computing default_int None	
Correct value 0	1 pt
Computing default_int (Some 9)	
Correct value 9	1 pt
Computing default_int None	
Correct value 0	1 pt
✓ Exercise 3: merge	Completed, 10 pts
Found merge with compatible type.	
Computing merge None (Some 3)	
Correct value (Some 3)	1 pt
Computing merge (Some 1) None	
Correct value (Some 1)	1 pt
Computing merge None None	
Correct value None	1 pt
Computing merge (Some 4) (Some 3)	
Correct value (Some 7)	1 pt
Computing merge None (Some 5)	
Correct value (Some 5)	1 pt
Computing merge (Some 4) (Some 1)	
Correct value (Some 5)	1 pt
Computing merge (Some 7) None	
Correct value (Some 7)	1 pt
Computing merge (Some 8) (Some 4)	
Correct value (Some 12)	1 pt
Computing merge (Some 4) (Some 10)	
Correct value (Some 14)	1 pt
Computing merge None (Some 10)	
Correct value (Some 10)	1 pt



Rechercher un cours



Mentions légales



POWERED BY
OPENedX