



- ▶ Introduction and overview
- ▶ Basic types, definitions and functions
- ▶ Basic data structures
- ▶ More advanced data structures
- ▼ Higher order functions

Table of Contents

Functional Expressions

Week 4 Échéance le déc 12, 2016 at 23:30 UTC

Functions as First-Class Values

Week 4 Échéance le déc 12, 2016 at 23:30 UTC

Functions with Multiple Arguments

Week 4 Échéance le déc 12, 2016 at 23:30 UTC

Partial Function Application

Week 4 Échéance le déc 12, 2016 at 23:30 UTC

Mapping Functions on Lists

Week 4 Échéance le déc 12, 2016 at 23:30 UTC

Folding Functions on Lists

Week 4 Échéance le déc 12, 2016 at 23:30 UTC

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

FUNCTIONS RETURNING FUNCTIONS (25/25 points)

The following function checks the pairwise equality of the elements of two lists, on the common length of both lists:

```
let rec equal_on_common l1 l2 = match l1,l2 with
| [],_ -> true
| _,[] -> true
| h1::r1,h2::r2 -> h1=h2 && equal_on_common r1 r2
```

1. Rewrite `equal_on_common : 'a list -> 'a list -> bool` by using nested `function .. ->` constructions. Using the `match .. with` construction or tuple patterns is forbidden. You can (and must) only call the operators `&&` and `=`, and the function `equal_on_common` recursively.

YOUR OCAML ENVIRONMENT

```
1 let rec equal_on_common =
2   function
3   | [] -> (function _ -> true)
4   | h1::r1 -> function
5     | [] -> true
6     | h2::r2 -> h1=h2 && equal_on_common r1 r2
7 ;;
8
```

Evaluate >

Switch >>

Typecheck

Reset Templ

Full-screen |

Check & Sa

Exercise complete (click for details)

25 pts

v Exercise 1: equal_on_common

Completed, 20 pts

Found equal_on_common with compatible type.

Computing equal_on_common ["#44560Camlba- "; "-"] ["be"; "be0CP"]

Correct value false

1 pt

Computing equal_on_common ["4456//, 0CP"; "#44560Caml"] ["4456//, 0CP"; "#44560Caml"]

Correct value true

1 pt

Computing

```
equal_on_common
["beba#0CP0CP , be"; "ba0Caml0CP#0CP0CP4456be4456"; "0Caml4456#babe";
"0Camlbe#//0CP"; "#, "]
["beba#0CP0CP , be"; "ba0Caml0CP#0CP0CP4456be4456"; "0Caml4456#babe";
"0Camlbe#//0CP"; "#, "; "4456 be"; ", "; ""; "-#"; "baba0CP -4456ba# "]
```

Correct value true

1 pt

Computing

```
equal_on_common
["0CP--0CP"; " - bebe0Caml-"; ""]
["0CP--0CP"; " - bebe0Caml-"; ""]
```

Correct value true

1 pt

Computing

```
equal_on_common
["be"; ""; "babe0Caml0Caml, "; "#be//"; "bebe0CP, "; "";
"4456baba0CPbe 4456"; "44564456"]
["be"; ""; "babe0Caml0Caml, "]
```

Correct value true

1 pt

Computing

Correct value false	1 pt
Computing equal_on_common [""; "OCaml#44560Caml#OCaml44560CP-"; "OCamlOCamlba#0CPbe"; "-, "; ""; " "] [""; "OCaml#44560Caml#OCaml44560CP-"; "OCamlOCamlba#0CPbe"; "0CPbe"; "//4456ba, 0CP-"; "0CP-bebebe44560Caml-"; ", 4456, , 0Caml"]	1 pt
Correct value false	1 pt
Computing equal_on_common ["be "; "0CP0CP0CPba"; "babe//4456//-//"; ""; ""; "//#- #//-"] ["be "; "0CP0CP0CPba"; "babe//4456//-//"]	1 pt
Correct value true	1 pt
Computing equal_on_common ["", //-, # -"; "-, 0CPbe, "; "0CP0CP-, ba"] ["", //-, # -"; "-, 0CPbe, "; "0CP0CP-, ba"; "ba0CP-- 0Camlbe0CP"; "ba//ba, 0Caml0CP-"; " ba0Caml"; "ba-0CP"; "##0CP#44564456be0CPbe"]	1 pt
Correct value true	1 pt
Computing equal_on_common ["ba"; "0CP"; ", 0Caml-"; "##4456be, 4456"; ", //, 4456-"; "ba0Camlba0Caml, ba0CPbe0CP"; "be0Caml, #44560Caml "] ["ba"; "0CP"; ", 0Caml-"; "##4456be, 4456"; ", //, 4456-"; "//4456, 0CP"; "44560Camlbe0Caml0Caml#0CP"; "#be# be"]	1 pt
Correct value false	1 pt
Found equal_on_common with compatible type.	
Computing equal_on_common [3; -3; -4; 2; 1] [3; -3; -4; 2; 1]	
Correct value true	1 pt
Computing equal_on_common [2; 2; 2; -4; -2] [2; 2]	1 pt
Correct value true	1 pt
Computing equal_on_common [4; -3; 0; -1; -3] [4; -3; 2; -4; -3]	1 pt
Correct value false	1 pt
Computing equal_on_common [-2; 2; -5; -4; -3] [-2; 2; -5; -4; -3; 0; 3; 4]	1 pt
Correct value true	1 pt
Computing equal_on_common [0; 3] [3; 3]	1 pt
Correct value false	1 pt
Computing equal_on_common [3; -1; 1; -4; -3; -4] [3; -1; 1; 3; -1; -1; 4]	1 pt
Correct value false	1 pt
Computing equal_on_common [-3; 2; -3; -1; 2; 2; 2] [-3; 2]	1 pt
Correct value true	1 pt
Computing equal_on_common [-1; 2] [-1; 2]	1 pt
Correct value true	1 pt
Computing equal_on_common [-2; -3; 4] [3; 3; -2]	1 pt
Correct value false	1 pt
Computing equal_on_common [1; 1] [1; 1; 2; 2; 1]	1 pt
Correct value true	1 pt
✓ Exercise 2: using nested <i>function</i>	Completed, 5 pts
Found a toplevel definition for equal_on_common .	
Your function has the expected shape!	5 pts



Rechercher un cours

