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

```
let rec equal_on_common =
  function
   | [] -> (function _ -> true)
   | h1::r1 -> function
       |[] -> true
       | h2::r2 -> h1=h2 && equal_on_common r1 r2
;;
```

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 ["#4456OCamlba- "; "-"] ["be"; "beOCP"]`

Correct value `false` 1 pt

Computing `equal_on_common ["4456//, OCP"; "#4456OCaml"] ["4456//, OCP"; "#4456OCaml"]`

Correct value `true` 1 pt

Computing
```
equal_on_common
  ["beba#OCPOCP , be"; "baOCamlOCP#OCPOCP4456be4456"; "OCaml4456#babe";
   "OCamlbe#//OCP"; "#, "]
  ["beba#OCPOCP , be"; "baOCamlOCP#OCPOCP4456be4456"; "OCaml4456#babe";
   "OCamlbe#//OCP"; "#, "; "4456 be"; ", "; ""; "-#"; "babaOCP -4456ba# "]
```
Correct value `true` 1 pt

Computing
```
equal_on_common
  ["OCP--OCP"; " - bebeOCaml-"; ""]
  ["OCP--OCP"; " - bebeOCaml-"; ""]
```
Correct value `true` 1 pt

Computing
```
equal_on_common
  ["be"; ""; "babeOCamlOCaml, "; "#be//"; "bebeOCP, "; "";
   "4456babaOCPbe 4456"; "44564456"]
  ["be"; ""; "babeOCamlOCaml, "]
```
Correct value `true` 1 pt

Computing

```
Correct value false                                                    1 pt
Computing
  equal_on_common
    ["", "OCaml#4456OCaml#OCaml4456OCP-"; "OCamlOCamlba#OCPbe"; "-, "; ""; " "]
    ["", "OCaml#4456OCaml#OCaml4456OCP-"; "OCamlOCamlba#OCPbe"; "OCPbe";
     "//4456ba, OCP-"; "OCP-bebebe4456OCaml-"; ",  4456, , OCaml"]
Correct value false                                                    1 pt
Computing
  equal_on_common
    ["be "; "OCPOCPOCPba"; "babe//4456//-//"; ""; ""; "//#- #//-"]
    ["be "; "OCPOCPOCPba"; "babe//4456//-//"]
Correct value true                                                     1 pt
Computing
  equal_on_common
    [", //-, # -"; "-, OCPbe,  "; "OCPOCP-, ba"]
    [", //-, # -"; "-, OCPbe,  "; "OCPOCP-, ba"; "baOCP--  OCamlbeOCP";
     "ba//ba, OCamlOCP-"; " baOCaml"; "ba-OCP"; "##OCP#44564456beOCPbe"]
Correct value true                                                     1 pt
Computing
  equal_on_common
    ["ba"; "OCP"; ", OCaml-"; "##4456be, 4456"; ", //, 4456-";
     "baOCamlbaOCaml, baOCPbeOCP"; "beOCaml, #4456OCaml "]
    ["ba"; "OCP"; ", OCaml-"; "##4456be, 4456"; ", //, 4456-"; "//4456, OCP";
     "4456OCamlbeOCamlOCaml#OCP"; "#be# be"]
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]
Correct value true                                                     1 pt
Computing equal_on_common [4; -3; 0; -1; -3] [4; -3; 2; -4; -3]
Correct value false                                                    1 pt
Computing equal_on_common [-2; 2; -5; -4; -3] [-2; 2; -5; -4; -3; 0; 3; 4]
Correct value true                                                     1 pt
Computing equal_on_common [0; 3] [3; 3]
Correct value false                                                    1 pt
Computing equal_on_common [3; -1; 1; -4; -3; -4] [3; -1; 1; 3; -1; -1; 4]
Correct value false                                                    1 pt
Computing equal_on_common [-3; 2; -3; -1; 2; 2; 2] [-3; 2]
Correct value true                                                     1 pt
Computing equal_on_common [-1; 2] [-1; 2]
Correct value true                                                     1 pt
Computing equal_on_common [-2; -3; 4] [3; 3; -2]
Correct value false                                                    1 pt
Computing equal_on_common [1; 1] [1; 1; 2; 2; 1]
Correct value true                                                     1 pt
```

v Exercise 2: using nested *function* Completed, 5 pts

```
Found a toplevel definition for equal_on_common .
Your function has the expected shape!                                  5 pts
```