USING FIRST CLASS FUNCTIONS (20/20 points)

1. Write a function `compose : ('a -> 'a) list -> ('a -> 'a)` that takes as argument a list `l` of functions, and that returns the function that is the composition of the functions in `l`. For instance, `compose [f;g;h] x = f (g (h x))`. Or with concrete functions, `compose [(fun x -> x+1);(fun x -> 3*x);(fun x -> x-1)] 2 = 4`.
2. Write a function `fixedpoint : (float -> float) -> float -> float -> float` that takes a function `f` of type `float -> float` and two floating-point arguments `start` and `delta`. The function `fixedpoint` applies repetitively `f` to the result of its previous application, starting from `start`, until it reaches a value `y` where the difference between `y` and `(f y)` is smaller than delta. In that case it returns the value of `y`. For instance, `fixedpoint cos 0. 0.001` yields approximately `0.739` (ref).

THE GIVEN PRELUDE

```
type int_ff = int -> int
```

YOUR OCAML ENVIRONMENT

```
let rec compose = function
  | [] -> (function x-> x)
  | f :: fs -> (function x-> f(compose fs(x)))
;;


let rec fixedpoint f start delta =
  if ((f(start) -. start) < delta && (f(start) -. start) >= 0.) ||
     ((start -. f(start)) < delta && (start -. f(start)) >= 0.) then start else
    fixedpoint f (f(start)) delta
;;
```

Evaluate >>

Switch >>

Typecheck

Reset Template

Full-screen [+]

Check & Save

Exercise complete (click for details) 20 pts

v Exercise 1: compose Completed, 10 pts

Found `compose` with compatible type.

Computing `compose [((-) 7)] 3`

Correct value `4` 1 pt

Computing `compose [((/) 4); ((/) 4); ((+) 10)] 1`

Correct exception `Division_by_zero` 1 pt

```
Computing
  compose
    [((+) 10); ((-) 7); ((-) 7); ((/) 4); ((/) 4); ((+) 10); ((-) 7); ((+) 10)]
    2
```

Correct exception `Division_by_zero` 1 pt

Computing `compose [((+) 10); ((/) 4); ((-) 7); ((-) 7)] 0`

Correct exception `Division_by_zero` 1 pt

Computing `compose [((+) 10)] -3`

Correct value `7` 1 pt

Found `compose` with compatible type.

```
Computing
  compose
    [String.uppercase; (fun s -> s ^ s); ((^) "@"); ((^) "@");
     (fun s -> s ^ s)]
    "OCaml4456,  -OCamlba, "
```

Correct value 1 pt
`"@@OCAML4456, -OCAMLBA, OCAML4456, -OCAMLBA, @@OCAML4456, -OCAMLBA, OCAML4456, -OCAMLBA, "`

Computing

```
Correct value "@@, OCAML, --, BA@@, OCAML, --, BA"                                  1 pt
Computing
  compose
    [(fun s -> s ^ s); ((^) "@"); (fun s -> s ^ s); String.uppercase;
     (fun s -> s ^ s); String.uppercase; ((^) "@"); (fun s -> s ^ s)]
    "#-OCaml-baOCamlOCaml# "
Correct value                                                                        1 pt
"@@#-OCAML-BAOCAMLOCAML# #-OCAML-BAOCAMLOCAML# @#-OCAML-BAOCAMLOCAML# #-OCAML-BAOCAMLOCAML#
Computing
  compose
    [((^) "@"); String.uppercase; String.uppercase; ((^) "@");
     (fun s -> s ^ s)]
    ""
Correct value "@@"                                                                   1 pt
Computing compose [((^) "@")] ", be#-bebebe"
Correct value "@, be#-bebebe"                                                        1 pt
v Exercise 2: fixedpoint                                              Completed, 10 pts
Found fixedpoint with compatible type.
Computing fixedpoint cos -0.473208262136879831 0.090277395580894
Correct value 0.690638749814055597                                                   1 pt
Computing fixedpoint ((*.) 0.1) 4.51160283640771276 0.085029684214584289
Correct value 0.045116028364077132                                                   1 pt
Computing fixedpoint sin -1.92564908707622928 0.0922694588259599
Correct value -0.806198067380910488                                                  1 pt
Computing fixedpoint sin -2.09632259472163751 0.0793876117246528901
Correct value -0.761134212758903805                                                  1 pt
Computing fixedpoint (fun _ -> 10.) 1.67745728601181554 0.0374015938242845 75
Correct value 10.                                                                    1 pt
Computing fixedpoint (fun _ -> 10.) -2.36680272848106 0.0956847110295913905
Correct value 10.                                                                    1 pt
Computing fixedpoint ((*.) 0.1) 0.0514588315747035452 0.0922058753114132906
Correct value 0.0514588315747035452                                                  1 pt
Computing fixedpoint cos 1.01902460345374291 0.0278029055361895955
Correct value 0.751208801119032432                                                   1 pt
Computing fixedpoint sin 4.57825069050975841 0.0824832718981356738
Correct value -0.742358412855530281                                                  1 pt
Computing fixedpoint (fun _ -> 10.) 0.530627709945003 19 0.0128325888287645368
Correct value 10.                                                                    1 pt
```