## USING FOLD TO PRODUCE LISTS (30/30 points)

The idea of this exercise is to write functions that iterate on lists, using the `fold_left` and `fold_right` functions from the `List` module.

1. Write a function `filter : ('a -> bool) -> 'a list -> 'a list` that takes a predicate `p` (a function returning a boolean) and a list `l` and returns all the elements of `l` that satisfy `p` (for which `p` returns `true`).
2. Define, using `List.fold_right`, a function `partition : ('a -> bool) -> 'a list -> 'a list * 'a list` that takes a predicate `p` and a list `l`, and that partitions `l` by `p`. It returns a pair of two lists `(lpos,lneg)`, where `lpos` is the list of all elements of `l` that satisfy `p`, and `lneg` is the list of all elements that do not satisfy `p`.
3. One way of sorting a list is as follows:
   - The empty list is already sorted.
   - If the list `l` has a head `h` and a rest `r`, then a sorted version of `l` can be obtained in three parts:
     1. first a sorted version of all elements of `r` that are smaller or equal to `h`,
     2. then `h`,
     3. then a sorted version of all elements of `r` that are greater than `h`.

   Write a function `sort:'a list-> 'a list` that implements this algorithm, using the function partition of the previous question. This sorting algorithm is also known as Quicksort where the pivot is always the first element of the list.

## YOUR OCAML ENVIRONMENT

```
let filter p l = List.fold_left
    (fun liste element -> if p element then element::liste else liste)
    []
    l
;;

let partition p l = List.fold_right
    (fun element (l1,l2) -> if p element then ((element::l1), l2) else (l1, (element::l2)))
    l
    ([], [])
;;

let rec sort = function
  | [] -> []
  | h::r -> let (l1,l2) = partition (function x -> x < h) r in
      sort(l1)@(h::sort(l2))
;;

```

Evaluate >

Switch >>

Typecheck

Reset Templ

Full-screen

Check & Sa

**Exercise complete (click for details)** 30 pts

v Exercise 1: `filter` Completed, 10 pts

Found `filter` with compatible type.

Computing `filter ((<>) 3) [3; -1; -5; -4; 4; -4; 4; 3; -3]`

Correct value `[-3; 4; -4; 4; -4; -5; -1]` 1 pt

Computing `filter ((<>) 3) [-3; 0; 2; -2; 3; 4; -4]`

Correct value `[-4; 4; -2; 2; 0; -3]` 1 pt

Computing `filter ((=) 0) [2; 2]`

Correct value `[]` 1 pt

Correct value `[]` 1 pt

Found `filter` with compatible type.

Computing `filter (fun f -> (f /. 2.) < 2.) []`

Correct value `[]` 1 pt

Computing `filter ((>) (-.1.)) []`

Correct value `[]` 1 pt

Computing
```
filter
  (fun f -> sin f > 0.)
  [4.98117835734239; 2.34792815881959527; -0.25822906947669555;
   3.70928389135566938; 4.2539182420509789; 3.15544846023830772;
   4.27225686686171713; 2.54638874021914674; -0.541083345971202334]
```
Correct value `[2.54638874021914674; 2.34792815881959527]` 1 pt

Computing
```
filter
  (fun f -> sin f > 0.)
  [-4.08470932477100401; -4.81173925898947097; -1.68037948677194615;
   -1.40902048880028641; 2.34094973065559842; 0.723296215382847;
   -4.39350545010528215]
```
Correct value 1 pt
```
[-4.39350545010528215; 0.723296215382847; 2.34094973065559842;
 -4.81173925898947097; -4.08470932477100401]
```
Computing
```
filter
  ((>) (-.1.))
  [1.52542012403445337; -4.73185337984949239; 4.04627853300372387;
   0.255844907756058504; -2.68039154118595224; 3.84086992455272913]
```
Correct value `[-2.68039154118595224; -4.73185337984949239]` 1 pt

## Exercise 2: `partition` — Completed, 10 pts

Found `partition` with compatible type.

Computing `partition (fun x -> x mod 2 = 0) []`

Correct value `([], [])` 1 pt

Computing `partition ((=) 0) [-4; 4; -4; -4]`

Correct value `([], [-4; 4; -4; -4])` 1 pt

Computing `partition (fun x -> x mod 2 = 0) [1]`

Correct value `([], [1])` 1 pt

Computing `partition ((<>) 3) [-2; -5; 1; -1; 0; 3]`

Correct value `([-2; -5; 1; -1; 0], [3])` 1 pt

Computing `partition (fun x -> x mod 2 = 0) [-1; -1; 0; 1; -1]`

Correct value `([0], [-1; -1; 1; -1])` 1 pt

Found `partition` with compatible type.

Computing
```
partition
  (fun f -> (f /. 2.) < 2.)
  [-3.72843577345909072; -4.01599291472169195; -3.07625365886337931;
   -2.6578386148361286; -0.40578772232387017; 4.96499423020131658;
   -1.44559454569663082; -1.9828952349090665; -1.44861508114946691;
   1.80348700873560386]
```
Correct value 1 pt
```
([-3.72843577345909072; -4.01599291472169195; -3.07625365886337931;
  -2.6578386148361286; -0.40578772232387017; -1.44559454569663082;
  -1.9828952349090665; -1.44861508114946691; 1.80348700873560386],
 [4.96499423020131658])
```
Computing
```
partition
  ((>) (-.1.))
  [4.47974523906124; 2.73108026004728277; -0.598154133713123315;
   1.36624396555211103; 4.63704171300947721; 1.67436645324364;
   -2.02393453677918256; -2.39072918037221793; -2.68397999011597221]
```
Correct value 1 pt
```
([-2.02393453677918256; -2.39072918037221793; -2.68397999011597221],
 [4.47974523906124; 2.73108026004728277; -0.598154133713123315;
  1.36624396555211103; 4.63704171300947721; 1.67436645324364])
```
Computing
```
partition
  (fun f -> (f /. 2.) < 2.)
  [-2.87077225184034956; 2.61972120531091424]
```
Correct value `([-2.87077225184034956; 2.61972120531091424], [])` 1 pt

Computing
```
partition
  (fun f -> sin f > 0.)
  [-0.124687907040522461; 1.51117606150621242; -4.81227346241108567;
   0.984117773014272501; -2.51078304764692728; -0.320907897581265367]
```
Correct value 1 pt
```
([1.51117606150621242; -4.81227346241108567; 0.984117773014272501],
 [-0.124687907040522461; -2.51078304764692728; -0.320907897581265367])
```
Computing

```
Correct value                                                          1 pt
  ([-4.56392609120229853; -0.438626113683580954; -2.15506007286676127;
    -4.67396250401961133],
   [4.73634483839143527])
```

v Exercise 3: `sort` — Completed, 10 pts

```
Found sort with compatible type.
Computing sort [2; -5; -5; -5; -2; 2; -5; -3; 3]
Correct value [-5; -5; -5; -5; -3; -2; 2; 2; 3]                        1 pt
Computing sort [2; 4; 1; -4; -3; -3]
Correct value [-4; -3; -3; 1; 2; 4]                                    1 pt
Computing sort [4; -2; 0; 3; 3]
Correct value [-2; 0; 3; 3; 4]                                         1 pt
Computing sort [1; 4; -2]
Correct value [-2; 1; 4]                                               1 pt
Computing sort [-5; 4; 3; 1; -4; -1; 0; 3]
Correct value [-5; -4; -1; 0; 1; 3; 3; 4]                              1 pt
Found sort with compatible type.
Computing sort [3.99674818195155268]
Correct value [3.99674818195155268]                                    1 pt
Computing
  sort
    [-3.94123270582666763; 1.35751286093912427; 2.99281950919087691;
     -0.0843253455040127164]
Correct value                                                          1 pt
  [-3.94123270582666763; -0.0843253455040127164; 1.35751286093912427;
   2.99281950919087691]
Computing
  sort
    [-0.609334859995460221; 0.791979896481613821; 4.53318859767687776;
     1.13598284683785078]
Correct value                                                          1 pt
  [-0.609334859995460221; 0.791979896481613821; 1.13598284683785078;
   4.53318859767687776]
Computing
  sort
    [3.96671418886582572; -3.37307780601918727; 2.66718496102002334;
     -3.35702044638636821; 0.195855919968855652; -4.64979342071091;
     -0.977577952609507]
Correct value                                                          1 pt
  [-4.64979342071091; -3.37307780601918727; -3.35702044638636821;
   -0.977577952609507; 0.195855919968855652; 2.66718496102002334;
   3.96671418886582572]
Computing
  sort
    [-4.57785945778382253; -0.951545814561200132; -1.42462652149533753;
     4.24778606643659451]
Correct value                                                          1 pt
  [-4.57785945778382253; -1.42462652149533753; -0.951545814561200132;
   4.24778606643659451]
```