❑ **Exercice 1** : *Représentation des ensembles d'entiers*

En OCaml, on propose de représenter un ensemble d'entiers, par la liste *triée* (dans l'ordre croissant) de ses éléments. Par exemple l'ensemble `{2; 3; 5; 7}` sera représenté par la liste `[2; 3; 5; 7]`. Ainsi une liste d'entiers représente correctement un ensemble lorsque :

— ses éléments sont dans l'ordre croissant,

— et chaque élément figure en un seul exemplaire.

Par exemple, les listes `[2; 3; 3; 5; 7]` (élément en double) ou `[2; 5; 3; 7]` (non triée) ne représentent pas correctement un ensemble.

1. Ecrire une fonction `est_ensemble: int list -> bool` qui renvoie `true` lorsque la liste d'entier fournie en argument représente correctement un ensemble.

```
let rec est_ensemble l =
  match l with
  | [] -> true
  | h::[] -> true
  | h1::h2::t -> h1<h2 && est_ensemble (h2::t);;
```

2. Ecrire une fonction `appartient: int -> int list -> bool` qui prend en argument un entier et une liste (représentant un ensemble) et renvoie `true` lorsque l'entier appartient à l'ensemble représenté par la liste. Par exemple `appartient 3 [2; 3; 5; 7]` renvoie `true`.

```
let rec appartient x l =
  match l with
  | [] -> false
  | h::t -> if h=x then true else
    if h>x then false else appartient x t;;
```

3. Ecrire une fonction `union : int list -> int list -> int list` qui prend en argument deux listes d'entiers (en supposant que ces deux listes représentent correctement des ensembles) et renvoie la liste d'entiers représentant l'union de ces deux listes. Par exemple `union [2; 5; 7] [5; 6; 7; 10];;` renvoie `[2; 5; 6; 7; 10]`.

```
let rec union l1 l2 =
  match l1, l2 with
  | [], l2 -> l2
  | l1, [] -> l1
  | h1::t1, h2::t2 -> if h1<h2 then h1::(union t1 l2) else
    if h1=h2 then h1::(union t1 t2) else
      h2::(union l1 t2);;
```