Option Informatique en Sup MPSI
Deux exercices

Itération d'une fonction

Question 1 • Rédigez en Caml une fonction :

```
itere : ('a -> 'a) -> int -> 'a -> 'a
```

spécifiée comme suit : `itere f n x` calcule l'image de x par la n-ième itérée de f. Vous lèverez une exception si $n < 0$.

Ensembles

▶ Nous nous proposons de gérer en Caml des ensembles (finis). Pour l'utilisateur, le type ensemble est un *type abstrait de données* : la façon dont ce type est effectivement implanté ne lui est pas connue. Seuls lui sont accessibles les signatures des fonctions permettant de construire et de consulter des ensembles :

```
ensemble_vide : unit -> 'a ensemble
ensemble_of_list : 'a list -> 'a ensemble
est_vide : 'a ensemble -> bool
appartient_a : 'a -> 'a ensemble -> bool
ajoute : 'a -> 'a ensemble -> 'a ensemble
enleve : 'a -> 'a ensemble -> 'a ensemble
union : 'a ensemble -> 'a ensemble -> 'a ensemble
intersection : 'a ensemble -> 'a ensemble -> 'a ensemble
inclus_dans : 'a ensemble -> 'a ensemble -> bool
egalite_ensembliste : 'a ensemble -> 'a ensemble -> bool
```

▶ Voici par exemple deux façons de construire l'ensemble $\{1, 2, 5\}$:

```
let e = ajoute 5 (ajoute 1 (ajoute 2 (ensemble_vide()))) ;;
let e = ensemble_of_list [2;1;5] ;;
```

Cet exemple sert aussi à vous rappeler que l'ordre d'énumération des éléments d'un ensemble est indifférent !

▶ Notez bien que, dans un ensemble, un même élément apparaît une fois au plus ; cette contrainte doit être satisfaite par les fonctions que vous écrirez !

▶ Autre contrainte : le coût (exprimé en nombre *Cons*) de l'application d'une fonction doit être un $\mathcal{O}$ de la taille de l'ensemble (ou, pour les opérations faisant intervenir deux ensembles, un $\mathcal{O}$ du produit des tailles des deux ensembles).

Question 1 Définissons le type `'a ensemble` comme suit :

```
type 'a ensemble = Set of 'a list;;
```

Rédigez les fonctions dont les signatures ont été décrites plus haut.

FIN