Les listes Caml

Durée : 1 heure

Question 1. Rédiger une fonction **premiers** qui prend en argument un entier n et une liste ℓ et qui renvoie la liste des n premiers éléments de ℓ (ou la liste ℓ tout entier si cette dernière comporte moins de n éléments).

```
premiers : int -> 'a list -> 'a list
```

Question 2. Déterminer le type et le rôle de la fonction suivante :

```
let rec flatten = function
  | []   -> []
  | t::q -> t @ (flatten q) ;;
```

puis en donner une définition utilisant la fonctionnelle **list_it**.

Question 3. Rédiger une fonction **itere** qui prend en arguments un entier n de type *int*, un élément x de type *'a* et une fonction f de type *'a -> 'a* et qui renvoie la liste $[x; f(x); \cdots; f^n(x)]$.

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

Question 4.

a) Rédiger une fonction **compose** qui prend en argument une liste de fonctions $[f_1; \cdots; f_n]$ de type *'a -> 'a* et renvoie la fonction $f_1 \circ f_2 \circ \cdots \circ f_n$. Lorsque la liste est vide cette fonction devra renvoyer la fonction identité. Un bonus est accordé si vous utilisez une fonctionnelle (**it_list** ou **list_it**).

```
compose : ('a -> 'a) list -> 'a -> 'a
```

b) Rédiger ensuite une fonction **compose_gauche** qui renvoie cette fois la fonction $f_n \circ f_{n-1} \circ \cdots f_1$.

```
compose_gauche : ('a -> 'a) list -> 'a -> 'a
```

Question 5. On appelle *sous-liste* de la liste $[x_1; x_2; \cdots; x_n]$ toute liste de la forme $[x_{i_1}; x_{i_2}; \cdots; x_{i_p}]$ où $1 \leqslant i_1 < i_2 < \cdots < i_p \leqslant n$. Rédiger une fonction **parts** qui prend en argument une liste et renvoie la liste de toutes ses sous-listes, dans un ordre quelconque.

```
parts : 'a list -> 'a list list
```

Question 6. Rédiger une fonction **somme** qui prend en arguments un entier n et une liste d'entiers ℓ et qui renvoie un booléen affirmant l'existence ou pas d'un sous-ensemble d'éléments de ℓ dont la somme est égale à n (on ne demande pas une fonction efficace).
Par exemple, **somme 10 [5; −7; 4; 8; 1]** doit renvoyer **true** puisque $10 = 5 + 4 + 1$ (ou encore $10 = 5 - 7 + 4 + 8$) alors que **somme 10 [3; −7; 4; 8; 1]** doit renvoyer **false**.

```
somme : int -> int list -> bool
```