

---

# TD2 : Algorithmes combinatoires

## 1 Parties d'un ensemble

### 1.1 Contrat

```
(* *****
parties : 'a list -> 'a list list
argument: l, une liste quelconque d'éléments
résultat: une liste de listes, dont les 2^(List.length l) éléments
          sont les sous-listes de l. Si les éléments de l sont
          différents, les listes éléments du résultat seront
          différentes aussi, et formeront les parties de l.
***** *)
```

### 1.2 Raffinage fonctionnel

#### ▷ Exercice 1

— Donner une formulation récursive comptant le nombre de parties d'un ensemble de cardinal  $n$ .

#### ▷ Exercice 2

- Écrire la fonction *ajout*, qui à partir d'un élément  $e$  et d'ensembles  $\{E_1, \dots, E_n\}$  renvoie l'ensemble  $\{E_1, \{e\} \cup E_1, \dots, E_n, \{e\} \cup E_n\}$ .
- Écrire la fonction *parties*, qui renvoie l'ensemble des parties d'un ensemble.

## 2 Permutations d'une liste

### 2.1 Contrat

```
(* *****
permutations : 'a list -> 'a list list
argument: l, une liste quelconque d'éléments
résultat: une liste de listes, dont les (List.length l)! éléments
          ont même longueur que l. Si les éléments de l sont
          différents, les listes éléments du résultat seront
          différentes aussi, et formeront les permutations de l.
***** *)
```

### 2.2 Raffinage fonctionnel

#### ▷ Exercice 3

— Donner une formulation récursive comptant le nombre de permutations d'un ensemble de taille  $n$ .

#### ▷ Exercice 4

- Écrire la fonction *insertions*, qui insère un élément à toutes les positions d'une liste.
- Écrire la fonction *permutations*, qui renvoie l'ensemble des permutations d'un ensemble.

---

## 3 Combinaisons

### 3.1 Contrat

Le contrat donne quelque chose comme :

```
(* *****  
combinaisons : 'a list -> int -> 'a list list  
argument: l, une liste quelconque d'éléments supposés différents  
          k, le nombre d'éléments distincts à tirer  
résultat: une liste de combinaisons. Chaque combinaison est elle-même  
          une liste d'éléments, dont les éléments  
          sont ceux de l.  
***** *)
```

### 3.2 Raffinage fonctionnel

- ▷ Exercice 5 Donner une formulation récursive comptant le nombre de combinaisons de  $k$  éléments d'un ensemble à  $n$  éléments.
- ▷ Exercice 6
  - Écrire la fonction *combinaisons* (contrat+code+tests)

# Algorithmes combinatoires

## I/ Parties d'un ensemble

### Exercice 1

Pour un ensemble à 0 elt on a 1 partie  $\{\emptyset\}$

Pour un ensemble à  $2^{n+1}$  elt on a  $2^{n+1}$  parties.  $2^n$  (ens. sans nouvel elt)  
 $2^n$  (ens. avec nouvel elt)

### Exercice 2:

let ajout e P = List.fold-right (fun xq → (e::x)::q) P [];;

let ajout e ens = List.map (fun qt → e::qt) @ ens;;

let parties P = List.fold-right ajout P []

1 [] → []

1 e::q → ajout e parties q;;

let parties P = List.fold-right (fun tqt → ajout t qt) P []

## II/ Permutations d'une liste

Exercice 3: si 0 elt 0 permutation  
sin elt n! permutation.

Exercice 4: let rec permutations P =

match P with

1 [] → []

1 e::q → List.flatten (List.map (insertion e) (permutations q));;

List.fold-right (fun tqt → List.flatten (List.map (insertion t) qt)) P [];;

@rec insertions<sup>P</sup> P =

match P with

1 [] → []