UE - Programmation Fonctionnelle

TD 2.

OCaml

Exercice 1 : 思考 : $\{e_1, \dots, e_n\} \leftrightarrow P(n) = 2^n$

e_i ✗

$P(n+1) = 2 \times P(n)$

$P(0) = 2^0 = 1$.

Exercice 2 : (1) let ~~ajout~~ rec ajout l elem =

match l with

~~| [] -> failwith~~

~~| [[]] -> [[]; [elem]]~~ ⇒ | [] -> [elem] []

| hd :: tl -> hd :: (elem :: hd) :: (ajout elem tl)

⇑ 考虑 List.flatten (List.map (fun partie_q -> [partie_q ; elem :: partie_q (除去)]) l).

Exercice 3 : (2) let rec parties l =

match l with

| [] -> [[]]

| hd :: tl -> ajout hd (parties tl)

---

let parties ens =

let rec aux ens acc = match ens with

| [] -> acc

| hd :: tl -> aux tl (ajout hd acc)

in aux ens [[]]

Exercice 3 : $P(n+1) = (n+1) \cdot P(n)$ $(n+1)! = (n+1) \times n!$ $0! = 1$

Exercice 4 : (* insertion : 'a -> 'a list -> 'a list list *)

(1) let rec insertion e l = match l with

| [] -> [[e]]

| hd :: tl -> (e :: l) :: (List.map (fun perm -> hd :: perm) (除去) (insertion e tl))

(2) let rec permutations l

match l with

| [] -> [[]]

| hd :: tl -> List.~~map~~ flatten (List.map (fun perm_q -> insertions perm_q) (permutation tl) (除去), tl)

tl

[1,2,3] [2,1,3] [2,3,1]

[1,3,2] [3,1,2] [3,2,1]

Exercice 5 : $\binom{n+1}{k+1} = \binom{n}{k} + \binom{n}{k+1}$; $\binom{n}{0} = 1$ $\binom{0}{k+1} = 0$

Exercice 6 :

```
let rec combinaisons k l =
    if k = 0 then [[]] else
        if List.length l = k then
            l
        else match l with
            | [] -> [[]]
            | hd :: tl -> List.map (fun liste -> hd :: liste)
(combinaison (k-1) tl) @ (combinaison k tl
```