Assistants de preuve

TD 4- Monades et Modules

1 Monades

1.1 Exceptions

On se donne la type des arbres binaires dont les feuilles sont étiquetées par des entiers naturels:

```
Inductive tree : Type :=
     Leaf : nat -> tree | Node : tree -> tree -> tree.
```

1- Écrire une fonction prenant en argument un arbre et retournant soit `Some n` où n est le produit de ses feuilles si aucune n'est égale à 0, ou alors `None` si une des feuilles est nulle.
2- Définir la monade d'exception. Outre le opérations habituelles des monades, elle comportera 2 opérations: `raise` pour lever une exception, et `try` pour la rattrapper. On utilisera le type `option` pour la coder.
3- Réécrire la fonction calculant le produit des feuilles d'un arbre en utilisant le mécanisme d'exception pour retourner 0 dès qu'une feuille est nulle.

1.2 Non déterminisme

Écrire la monade de non-déterminisme, qui permet d'exécuter de manière non déterministe une tâches parmi plusieurs. Outre le opérations habituelles des monades, elle comportera une opération `par` de type $M(A) \to M(A) \to M(A)$ telle que `par e1 e2` effectue de manière non déterministe soit `e1`, soit `e2`.

2 Modules

1- Écrire une signature de modules représentant un type muni d'un préordre.
2- Écrire une signature de module représentant un foncteur prenant un type muni d'un ordre (fonction booléenne), et produisant une structure d'ensembles finis implantant les fonctions suivantes: test d'appartenance, ensemble vide, ajout d'un élément, retrait d'un élément, ainsi que les propriétés caractéristiques de ces opérations.
3- Implanter cette signature de foncteur en utilisant une structure de liste pour représenter les ensembles finis.