

# Méthodes formelles de vérification (MF) TD n° 3 : Preuve de programmes fonctionnels

### Exercice 1:

Lesquelles des relations suivantes sont bien fondées?

```
1. (\mathbb{R}, <)
2. ((1/2, 1], <)
3. (\mathbb{N}, \leqslant)
4. (\mathbb{N}, <)
```

5. 
$$(\mathbb{Z}, >)$$

6. ( Nat, Lt )

7. (List[Nat], Smaller)

```
Smaller(1::ls ,[]) = false
où Smaller([] ,_) = true
Smaller(1::ls ,m::ms) = Smaller(ls,ms)
```

8. Donner une définition d'ordre bien fondé pour les naturels binaires définis au TD1 (Bin).

## Exercice 2:

On considère le type des arbres binaires de recherche (ABR) qui sont des arbres binaires (BinTree') respectant la propriété inductive suivante :

- 1. Spécifier la fonction Flatten (en utilisant Mem et Find) pour les ABR.
- 2. Prouver que l'implémentation de flatten donnée au TD1 satisfait la spécification (sinon corrigez-la et prouvez-la).

## Exercice 3:

Prouver la correction des fonctions Partition et FusionSort du TD2.

### Exercice 4:

Compléter les preuves laisses comme exercice au cours.

```
1. Etant donnée Append : List[*] -> List[*] -> List[*], où
   Spec_Append(11,12,1) =
         Append(11,12) = 1 \Rightarrow
             |1|=|11|+|12| \
             (\forall i \in \mathbb{N}, (0 \leqslant i \leqslant |11|) \Rightarrow 1[i] = 11[i]) \land
                       (\forall i \in \mathbb{N}, (0 \leqslant i \leqslant |12|) \Rightarrow 1[|11|+i] = 12[i])
   et
   Append([],1)
   Append(a::11,12) = a::(Append(11,12))
   prouver que :
   (\forall 11, 12, 1, Append(11,12) = 1 \Rightarrow Spec_Append(11,12,1))
2. Etant donnée Insert : * -> List[*] -> List[*], où
   Spec_Insert(a,l,l') = Insert(a,l) = l' \Rightarrow
                                Ordered(1') \land (Ms(1') = Sg(a) \cup Ms(1))
   et
   Insert(a,[]) = a::[]
   Insert(a,b::l) = if a \leq b then a::(b::l) else b::(Insert(a,l))
   prouver que :
   (\forall a, l, l', Insert(a,l) = l' \Rightarrow Spec_Insert(a,l,l')
3. Etant donnée qsort : List[*] -> List[*], où
   Spec_qsort(1,1') = qsort(1) = 1' \Rightarrow
                             Ordered(1') \land (Ms(1') = Sg(a) \cup Ms(1))
   et
   gsort([]) = []
   qsort(a::1) = let (11, 12) = split (a, 1) in
                        Append(qsort(11),(a::qsort(12))
```

avec split étant la fonction qui divise la liste donnée en deux, où la première contient les éléments plus petits que a, et la deuxième les autres.

- a. Donner une spécification et une implémentation pour split.
- b. En supposant la correction de split prouver que pour tout 1 et 1', Spec\_qsort(1,1')