Eléments de Corrections pour le concours 2018

ALSDD :

Arbre de recherche binaire pur

- Vérifier si un arbre binaire r est pur (ou strictement binaire), c-a-d vérifier que chaque nœud interne possède deux fils différents de nil :

```
Verif( r:ptr ) : booleen
SI ( r = nil )
        return vrai
SINON
        SI ( ( fg(r) = nil && fd(r) <> nil ) || ( fg(r) <> nil && fd(r) = nil ) )
                return faux
        SINON
                return ( Verif( fg(r) ) && Verif( fd(r) ) )
        FSI
FSI
```

- Transformer un arbre binaire r en un arbre binaire pur en rajoutant des nœuds (contenant des zéros). La fonction retourne le nombre nœuds rajoutés pour la transformation :

```
Transf( r:ptr ) : entier
SI ( r = nil )    return 0
SINON             SI ( ( fg(r) = nil && fd(r) <> nil ) )
                          Aff-fg( r , CreerNoeud(0) ) ;
                          return 1 + Transf( fd(r) )
                  SINON
                     SI ( fg(r) <> nil && fd(r) = nil ) )
                          Aff-fd( r , CreerNoeud(0) ) ;
                          return 1 + Transf( fg(r) )
                     SINON
                          return ( Transf( fg(r) ) + Transf( fd(r) ) )
                     FSI
                  FSI
FSI
```

SFSD :

Essai linéaire en mémoire centrale

- Algo de recherche et d'insertion :

Tab1[N] est un tableau de structures (elem :entier ; vide :bool)

```
Rech( x : entier var trouv:bool, var i:entier )
i ← h(x) ;
trouv ← faux ;
TQ ( Non trouv et Tab1[i].vide = faux )
        SI ( Tab1[i].elem = x ) trouv ← vrai
        SINON i ← i-1 ; SI ( i<0 ) i← N-1 FSI
        FSI
FTQ
```

NbIns est un entier global utilisé pour compter le nombre d'insertions réalisées (initialisé à 0)

```
Ins( x:entier )
Rech( x, trouv ,i ) ;
SI ( Non trouv && NbIns < N-1)
        NbIns++ ;
        Tab1[i].elem ← x ; Tab1[i].vide ← faux
FSI
```

- Cas où chaque case du tableau peut contenir 2 entiers (la taille du tableau est N = 7)
Les cases du tableau seront des structures (elm:tableau[2] d'entier, NB:entier)
Après l'insertion des valeurs suivantes : 22 , 0 , 45 , 1 , 8 , 3 , 10 , 6 , 13 , 14 , 21 , 5

Le tableau Tab1 sera :

	0	1	2	3	4	5	6
elm1	0	22	10	45	5	14	6
elm2	8	1		3		21	13
NB	2	2	1	2	1	2	2

Déroulement :

h(22) = 1	→ Tabl[1].elm1
h(0) = 0	→ Tabl[0].elm1
h(45)=3	→ Tabl[3].elm1
h(1)=1	→ Tabl[1].elm2
h(8)=1 (collision)	→ Tabl[0].elm2
h(3)=3	→ Tabl[3].elm2
h(10)=3 (collision)	→ Tabl[2].elm1
h(6)=6	→ Tabl[6].elm1
h(13)=6	→ Tabl[6].elm2
h(14)=0 (collision)	→ Tabl[5].elm1
h(21)=0 (collision)	→ Tabl[5].elm2
h(5)=5 (collision)	→ Tabl[4].elm1