Quelques exemples & le fonctionnement du programme

Le projet est fait individuellement en 2 jours seulement.

L'utilisation des types abstraits et les modules c'est pour faciliter l'implantation des arbres 2-3-4 avec n'importe quel ensemble ordonnée (L'ensemble ordonné des entiers, L'ensemble des caractère avec un ordre lexicale,….).

I - Création de deux ensembles d'entiers représenté par un arbre 2-3-4 :

Le premier ensemble : ens .

Le deuxième ensemble : ens2 .

```
val ens : Int234.a234 =
  Int234.Noeud3 ((23, 89),
   Int234.Noeud2 (3, Int234.Noeud2 (0, Int234.VideA234, Int234.VideA234),
    Int234.Noeud4 ((11, 12, 14), Int234.VideA234, Int234.VideA234,
     Int234.VideA234, Int234.VideA234)),
   Int234.Noeud4 ((49, 57, 70),
    Int234.Noeud2 (44, Int234.VideA234, Int234.VideA234),
    Int234.Noeud2 (55, Int234.VideA234, Int234.VideA234),
    Int234.Noeud3 ((60, 63), Int234.VideA234, Int234.VideA234,
     Int234.VideA234),
    Int234.Noeud2 (83, Int234.VideA234, Int234.VideA234)),
   Int234.Noeud2 (96, Int234.Noeud2 (93, Int234.VideA234, Int234.VideA234),
    Int234.Noeud2 (97, Int234.VideA234, Int234.VideA234)))
val ens2 : Int234.a234 =
  Int234.Noeud3 ((43, 59),
   Int234.Noeud3 ((18, 20), Int234.VideA234, Int234.VideA234,
    Int234.VideA234),
   Int234.Noeud3 ((46, 54), Int234.VideA234, Int234.VideA234,
    Int234.VideA234),
   Int234.Noeud4 ((70, 84, 99), Int234.VideA234, Int234.VideA234,
    Int234.VideA234, Int234.VideA234))
```

2 – On essaye de chercher la valeur 57 dans ens avec la fonction « search », le résultat retourné doit être VRAI car ens contient 57 (Ligne 5).

```
# Int234.search ens 57;;
- : bool = true
#
```

3 – On teste les fonctions de conversions a234_vers_abic et abic_vers_ a234 :
Pour cela on va transformer « ens » en un arbre bicolore qu'on note « x », en suite on transforme l'arbre « x » en un arbre 2-3-4 et on stocke le résultat dans une variable y et normalement si les deux fonctions marchent bien on doit avoir que ens = y.

```
# let x = Int234.a234_vers_abic ens;;
val x : Int234.ab = <abstr>
# let y = Int234.abic_vers_a234 x;;
val y : Int234.a234 =
  Int234.Noeud3 ((23, 89),
   Int234.Noeud2 (3, Int234.Noeud2 (0, Int234.VideA234, Int234.VideA234),
    Int234.Noeud4 ((11, 12, 14), Int234.VideA234, Int234.VideA234,
     Int234.VideA234, Int234.VideA234)),
   Int234.Noeud4 ((49, 57, 70),
    Int234.Noeud2 (44, Int234.VideA234, Int234.VideA234),
    Int234.Noeud2 (55, Int234.VideA234, Int234.VideA234),
    Int234.Noeud3 ((60, 63), Int234.VideA234, Int234.VideA234,
     Int234.VideA234),
    Int234.Noeud2 (83, Int234.VideA234, Int234.VideA234)),
   Int234.Noeud2 (96, Int234.Noeud2 (93, Int234.VideA234, Int234.VideA234),
    Int234.Noeud2 (97, Int234.VideA234, Int234.VideA234)))
# y = ens;;
- : bool = true
```

4 – On teste d'insérer la valeur 100 dans « ens » avec la fonction dédié à cet opération « insert » :

```
# let ens3 = Int234.insert 100 ens;;
val ens3 : Int234.a234 =
  Int234.Noeud3 ((23, 89),
   Int234.Noeud2 (3, Int234.Noeud2 (0, Int234.VideA234, Int234.VideA234),
    Int234.Noeud4 ((11, 12, 14), Int234.VideA234, Int234.VideA234,
     Int234.VideA234, Int234.VideA234)),
   Int234.Noeud4 ((49, 57, 70),
    Int234.Noeud2 (44, Int234.VideA234, Int234.VideA234),
    Int234.Noeud2 (55, Int234.VideA234, Int234.VideA234),
    Int234.Noeud3 ((60, 63), Int234.VideA234, Int234.VideA234,
     Int234.VideA234),
    Int234.Noeud2 (83, Int234.VideA234, Int234.VideA234)),
   Int234.Noeud2 (96, Int234.Noeud2 (93, Int234.VideA234, Int234.VideA234),
    Int234.Noeud3 ((97, 100), Int234.VideA234, Int234.VideA234,
     Int234.VideA234)))
# Int234.search ens3 100;;
- : bool = true
```

La valeur 100 est bien ajouté, et la fonction « search ens 100 » retourne vrai.

5 – La fonction « arbre_alea » génère un arbre 2-3-4 aléatoire et elle prend en paramètre la limite et le nombre d'éléments respectivement.
On teste de générer un arbre 2-3-4 aléatoire avec 20 éléments compris entre 0 et 50 .

```
# Int234.arbre_alea 50 20;;
- : Int234.a234 =
Int234.Noeud2 (24,
 Int234.Noeud3 ((8, 19),
  Int234.Noeud3 ((1, 2), Int234.VideA234, Int234.VideA234, Int234.VideA234),
  Int234.Noeud3 ((9, 15), Int234.VideA234, Int234.VideA234, Int234.VideA234),
  Int234.Noeud3 ((20, 22), Int234.VideA234, Int234.VideA234, Int234.VideA234)),
 Int234.Noeud3 ((32, 36),
  Int234.Noeud2 (26, Int234.VideA234, Int234.VideA234),
  Int234.Noeud3 ((33, 35), Int234.VideA234, Int234.VideA234, Int234.VideA234),
  Int234.Noeud4 ((38, 40, 41), Int234.VideA234, Int234.VideA234,
   Int234.VideA234, Int234.VideA234)))
```

6 – La fonction union qui prend en paramètre deux arbres 2-3-4 et retourne leur union.
On créer un arbre 2-3-4 qu'on nomme « un » et qui est l'union des deux arbres 2-3-4 « ens » et « ens2 ».

```
# let un = Int234.union ens ens2;;
val un : Int234.a234 =
  Int234.Noeud3 ((23, 89),
   Int234.Noeud3 ((3, 12),
    Int234.Noeud2 (0, Int234.VideA234, Int234.VideA234),
    Int234.Noeud2 (11, Int234.VideA234, Int234.VideA234),
    Int234.Noeud4 ((14, 18, 20), Int234.VideA234, Int234.VideA234,
     Int234.VideA234, Int234.VideA234)),
   Int234.Noeud2 (57,
    Int234.Noeud2 (49,
     Int234.Noeud4 ((43, 44, 46), Int234.VideA234, Int234.VideA234,
      Int234.VideA234, Int234.VideA234),
     Int234.Noeud3 ((54, 55), Int234.VideA234, Int234.VideA234,
      Int234.VideA234)),
    Int234.Noeud2 (70,
     Int234.Noeud4 ((59, 60, 63), Int234.VideA234, Int234.VideA234,
      Int234.VideA234, Int234.VideA234),
     Int234.Noeud3 ((83, 84), Int234.VideA234, Int234.VideA234,
      Int234.VideA234))),
   Int234.Noeud2 (96, Int234.Noeud2 (93, Int234.VideA234, Int234.VideA234),
    Int234.Noeud3 ((97, 99), Int234.VideA234, Int234.VideA234,
     Int234.VideA234)))
```

7 – La fonction d’intersection :
On teste l’intersection des deux arbres 2-3-4 « ens » et « ens2 » :

```
# Int234.intersection ens ens2;;
- : Int234.a234 = Int234.Noeud2 (70, Int234.VideA234, Int234.VideA234)
#
```

8 – La fonction de différence :
On teste la différence des deux arbres 2-3-4 « ens » et « ens2 » (difference ens ens2) , le résultat c’est l’arbre 2-3-4 qui contient les éléments qui sont dans « ens » mais pas dans « ens2 ».

```
# Int234.difference ens ens2;;
- : Int234.a234 =
Int234.Noeud3 ((49, 89),
 Int234.Noeud3 ((12, 23),
  Int234.Noeud4 ((0, 3, 11), Int234.VideA234, Int234.VideA234,
   Int234.VideA234, Int234.VideA234),
  Int234.Noeud2 (14, Int234.VideA234, Int234.VideA234),
  Int234.Noeud2 (44, Int234.VideA234, Int234.VideA234)),
 Int234.Noeud2 (57, Int234.Noeud2 (55, Int234.VideA234, Int234.VideA234),
  Int234.Noeud4 ((60, 63, 83), Int234.VideA234, Int234.VideA234,
   Int234.VideA234, Int234.VideA234)),
 Int234.Noeud2 (96, Int234.Noeud2 (93, Int234.VideA234, Int234.VideA234),
  Int234.Noeud2 (97, Int234.VideA234, Int234.VideA234)))
```

9 - La fonction de différence symétrique :
On teste la différence des deux arbres 2-3-4 « ens » et « ens2 » (difference_symetrique ens ens2) :

```
# Int234.difference_symetrique ens ens2;;
- : Int234.a234 =
Int234.Noeud3 ((49, 89),
 Int234.Noeud3 ((12, 23),
  Int234.Noeud4 ((0, 3, 11), Int234.VideA234, Int234.VideA234,
   Int234.VideA234, Int234.VideA234),
  Int234.Noeud4 ((14, 18, 20), Int234.VideA234, Int234.VideA234,
   Int234.VideA234, Int234.VideA234),
  Int234.Noeud4 ((43, 44, 46), Int234.VideA234, Int234.VideA234,
   Int234.VideA234, Int234.VideA234)),
 Int234.Noeud3 ((57, 63),
  Int234.Noeud3 ((54, 55), Int234.VideA234, Int234.VideA234, Int234.VideA234),
  Int234.Noeud3 ((59, 60), Int234.VideA234, Int234.VideA234, Int234.VideA234),
  Int234.Noeud3 ((83, 84), Int234.VideA234, Int234.VideA234, Int234.VideA234)),
 Int234.Noeud2 (96, Int234.Noeud2 (93, Int234.VideA234, Int234.VideA234),
  Int234.Noeud3 ((97, 99), Int234.VideA234, Int234.VideA234, Int234.VideA234)))
```

10 – La fonction d'inclusion qui teste l'inclusion d'un arbre 2-3-4 dans un autre.

11- La fonction d'égalité qui teste l'égalité entre deux arbre 2-3-4.

12 – La fonction « fold_f » et « fold_r » sur les arbres 2-3-4 qui sont équivalentes respectivement à « List.fold_left » et « List.fold_right » sur les listes.

13 – La fonction « cardinal_fold » et « cardinal » qui renvoient le cardinal de l'arbre 2-3-4 mis en paramètre.
La fonction « cardinal_fold » utilise les fonctions « fold_f » ou « fold_r ».
La fonction « cardinal » convertit l'arbre 2-3-4 entré en paramètre en arbre bicolore pour faciliter le calcul du cardinal avec aucun effet de bord sur l'arbre en question.

14 – La fonction de filtrage « filtrer_selon g e » :
Elle permet de filtrer un arbre 2-3-4 selon une fonction, elle retourne un nouvel arbre 2-3-4 contenant les éléments de l'arbre « e » qui vérifient la fonction « g » .

15 – La fonction de séparation « separation x e ».

Elle sépare l'arbre 2-3-4 « e » en deux, tel que le premier arbre 2-3-4 retourné contient que les éléments supérieurs strictement à x et le deuxième contient les éléments inférieurs strictement à x.

Elle retourne les deux arbres 2-3-4 calculés et un booléen entre ces deux derniers qui indique si l'élément x appartient à e ou non.

Exemple : On sépare l'arbre « ens » en deux : L'arbre « a1 » avec des éléments supérieurs strictement 20 et l'arbre « a2 » avec des éléments inférieurs strictement à 20.

```
# let a1,_,a2 = Int234.separation 20 ens;;
val a1 : Int234.a234 =
  Int234.Noeud2 (12,
   Int234.Noeud4 ((0, 3, 11), Int234.VideA234, Int234.VideA234,
    Int234.VideA234, Int234.VideA234),
   Int234.Noeud2 (14, Int234.VideA234, Int234.VideA234))
val a2 : Int234.a234 =
  Int234.Noeud2 (89,
   Int234.Noeud4 ((49, 57, 70),
    Int234.Noeud3 ((23, 44), Int234.VideA234, Int234.VideA234,
     Int234.VideA234),
    Int234.Noeud2 (55, Int234.VideA234, Int234.VideA234),
    Int234.Noeud3 ((60, 63), Int234.VideA234, Int234.VideA234,
     Int234.VideA234),
    Int234.Noeud2 (83, Int234.VideA234, Int234.VideA234)),
   Int234.Noeud2 (96, Int234.Noeud2 (93, Int234.VideA234, Int234.VideA234),
    Int234.Noeud2 (97, Int234.VideA234, Int234.VideA234)))
_
```