IUT de Bayonne - Département Informatique
DUT 2ème année – AP – Programmation Fonctionnelle

TD n°4

Vecteurs, Arbres binaires, Arbres binaires ordonnés

A.- Manipulation des vecteurs

1. Pour chacune des vecteurs suivants, écrivez l'expression qui les construit à l'aide de l'opérateur **`vector`**

```
1    #(un (bel) exemple)
2    #((((1) 2) 3) 4)
3    #( ((a) (b) (c)) g () )
4    #( ((ré mi) do) (fa la) )
5    #("unix" (is a) "trademark" (of) bell labs)
```

2. Coder (en Scheme) les fonctions classiques suivantes, après avoir écrit la relation de récurrence :

a) procédure prédéfinie **`append`** version simple :
Ecrire la fonction de concaténation à 2 arguments, qui place le vecteur M en fin de vecteur L.

b) procédure prédéfinie **`reverse`** :
Inverse les éléments d'un vecteur L quelconque, au premier niveau.

c) procédure **`suppression :`**
permet de supprimer un éléments s (de niveau 1) de toutes les occurrences contenues dans le vecteur L.

d) procédure **`fusion`** :
permet de fusionner deux vecteurs en les triants par ordre croissant de valeurs.
On supprime les doublons.

B.- Création d'arbres

1. **Dessiner** les arbres définis par les expressions ex1,… de l'encadré ci-dessous.
2. **Écrire** les expressions permettant de créer 'proprement' (c'est-à-dire à l'aide du constructeur d'arbres binaires) les arbres binaires ex3 et ex5 identiques respectivement à ex3-bis / ex3-ter et ex5-bis / ex5-ter / ex5-kar.
3. **Expliquer** la raison des équivalences entre les versions normalisées et les versions –bis et –ter, -kar lorsqu'elles existent.

C.-Procédures d'observatuion, de en Scheme les opérations récursives réalisant les services suivants :

1. Hauteur d'un arbre binaire quelconque
2. Parcours d'arbres binaires quelconques :
   a) Parcours infixe d'un arbre binaire quelconque
   b) Parcours préfixe d'un arbre binaire quelconque
   c) Parcours postfixe d'un arbre binaire quelconque
3. Recherche d'un élément E dans un arbre binaire quelconque
4. Recherche d'un élément E dans un arbre binaire ordonné
5. Ajout d'un élément dans un arbre binaire ordonné

Méthode :
On commencera par écrire d'abord la relation de récurrence.
Le codage en Scheme se fera dans un deuxième temps.

On pourra utiliser les arbres binaires suivants pour tester les fonctions.

```
(define ex1 (arbre2:cons 4
                         (arbre2:cons 8
                                 ()
                                 ()
                          )
                          (arbre2:cons 9
                                       (arbre2:cons 10
                                                    ()
                                                    ()
                                       )
                                       (arbre2:cons 15
                                                    ()
                                                    ()
                                        )
                          )
                )
)
```

```
(4 (8 () ())
   (9 (10 () ())
      (15 () ())
   )
)
```

arbre binaire **non** ordonné de nombres

```
(define ex1-bis (arbre2:cons 4
                             (list 8 () ())
                             (list 9
                                     (list 10 () ())
                                     (list 15 () ())
                              )
                  )
 )
```

```
(4 (8 () ())
   (9 (10 () ())
      (15 () ())
   )
)
```

arbre binaire **non** ordonné de nombres

```
(define ex2 (arbre2:cons
                  15
                  (arbre2:cons 10
                               (arbre2:cons 10 () ())
                               (arbre2:cons 12
                                            (arbre2:cons 11 () ())
                                            ()
                                )
                   )
                  (arbre2:cons 20 () ())
            )
)
```

```
(15
   (10
      (10 () ())
      (12
         (11 () ())
         ()
       )
    )
   (20 () ())
)
```

arbre binaire **ordonné** de nombres

```
(define ex3-bis '(a
                   (b (2 () ()) (d (5 () ()) ()))
                   (c () (3 () ()))
                 )
 )
```

```
(a
  (b
     (2 () ())
     (d
        (5 () ())
        ()
     )
  )
  (c
     ()
     (3 () ())
  )
)
```

arbre binaire **non** ordonné de symboles

```
(define ex3-ter (arbre2:cons 'a
                             (list 'b
                                   (list 2 () ())
                                   (list 'd
                                         (list 5 () ())
                                         ()
                                    )
                              )
                             (list 'c
                                   ()
                                   (list 3 () ())
                              )
```

```
(a
  (b
     (2 () ())
     (d
        (5 () ())
        ()
     )
  )
  (c
     ()
     (3 () ())
  )
)
```

arbre binaire **non** ordonné de symboles

)) (define **ex4** (arbre2:cons #\o (arbre2:cons #\j (arbre2:cons #\d (arbre2:cons #\d () ()) ()) (arbre2:cons #\m (arbre2:cons #\k () ()) ())) (arbre2:cons #\t (arbre2:cons #\p () ()) (arbre2:cons #\u () ()))))	(#\o (#\j (#\d (#\d () ()) ()) (#\m (#\k () ()) ())) (#\t (#\p () ()) (#\u () ()))) arbre binaire **ordonné** de caractères
(define **ex5-bis** '("miel" ("etoile" () ("faible" () ())) ("science" ("patron" () ()) ())))	=> ("miel" ("etoile" () ("faible" () ())) ("science" ("patron" () ()) ())) arbre binaire **ordonné** de chaînes de car.
(define **ex5-ter** (arbre2:cons "miel" '("etoile" () ("faible" () ())) '("science" ("patron" () ()) ())))	=> ("miel" ("etoile" () ("faible" () ())) ("science" ("patron" () ()) ())) arbre binaire **ordonné** de chaînes de car.
(define **ex5-kar** (list "miel" (list "etoile" () (list "faible" () ())) (list "science" (list "patron" () ()) ())))	=> ("miel" ("etoile" () ("faible" () ())) ("science" ("patron" () ()) ())) arbre binaire **ordonné** de chaînes de car.

Exercices complémentaires

6. Calcul du nombre de feuilles
7. Calcul du nombre de nœuds
8. Parcours de l'arbre en largeur