TP numéro 6

Fonctions en argument

• Écrire une fonction qui applique une fonction unaire à tous les nombres d'une liste quelconque.

```
(applique-nombres sqr '(a 3 "ert" (t 5) 4 b)) → (a 9 "ert" (t 5) 16 b)
```

Abstraction sur les arbres

• En utilisant la fonction abstraite sur les arbres vue en cours, écrire une nouvelle version des fonctions hauteur et miroir.

Map et apply

• Redéfinir la fonction map pour les fonctions binaires (comme applique-à-tous est la fonction map pour les fonctions unaires).

```
(map_binaire + '(1 2 3) '(4 5 6)) → (5 7 9)
```

Définir les fonctions suivantes en utilisant map et/ou apply

- Une fonction qui calcule les racines carrées des nombres d'une liste de nombres.

  ```
  (racines '(4 16 9)) → (2 4 3)
  ```

- Une fonction qui calcule la somme des racines carrées des nombres d'une liste de nombres.

  ```
  (somme-racines '(4 16 9)) → 9
  ```

- Une fonction qui calcule les racines carrées des nombres d'une liste de listes de nombres.

  ```
  (racines-listes '((4 16) (9 1 4))) → ((2 4) (3 1 2))
  ```

- Une fonction qui calcule la moyenne des nombres d'une liste de nombres.

  ```
  (moyenne '(4 6 13 1)) → 6
  ```

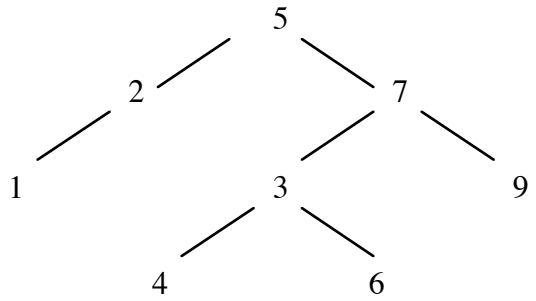
- La fonction applique-nombres définie ci-dessus.

- Une fonction qui calcule la somme des nombres d'une liste de listes de nombres.

  ```
  (somme '((1 2 3) (4 5) (6 7))) → 28
  ```

Révisions

On définit l'arbre a suivant :



• Définir une fonction qui, étant donné x un nombre et l une liste de couples (nombre, liste), retourne le deuxième élément d'un couple de l dont le premier élément est x.

```
(cherche 6 '((3 (a d r)) (6 (a z e)) (2 (e r)))) → (a z e)
```

• Définir une fonction qui, étant donné un nombre x et une liste de nombres l, construit deux listes : celle des nombres de l inférieurs ou égaux à x, et celle des nombres de l supérieurs à x.

```
(separe 3 '(5 1 2 3 6 4)) → ((1 2 3) (5 6 4))
```

• Définir une version de la fonction précédente qui traite la liste en profondeur.

```
(separe-prof 3 '(5 (6 (1 (2)) 4) 7)) → ((1 2) (5 6 4 7))
```

• Définir une fonction qui, étant donné un nombre x et un arbre de nombres a, construit la liste des valeurs de a supérieures à x.

```
(valeurs 3 a) → (5 7 4 6 9)
```

• Définir une fonction qui, étant donné une fonction f et un arbre a, applique la fonction f à chaque valeur de l'arbre a.

```
(applique-arbre sqr a)→(25(4(1()())())(49(9(16()())(36()()))(81()()))))
```

• Utiliser la fonction précédente pour définir une fonction qui enlève 1 à toutes les valeurs impaires d'un arbre d'entiers.

```
(rend-pair a) → (4(2(0()())())(6(2(4()())(6()()))(8()()))))
```

• Définir une fonction qui calcule le minimum d'une liste de listes de nombres.

```
(min-liste '((2 2 5) (1 6 3) (5 7))) → 7
```

• Définir une fonction qui fait la somme des seconds éléments des couples d'une liste de couples.

```
(ajoute2nd '((a 5) (b 7) (c 3))) → 15
```