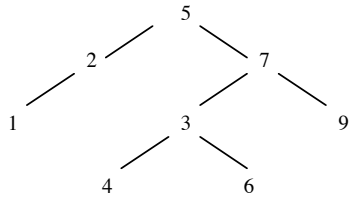
TP numéro 7

On définit l'arbre `a1` suivant :



Fonctions en argument

• Écrire une fonction qui applique une fonction unaire à chaque feuille d'un arbre.

```
(applique-feuilles (lambda (x) (* x 2)) a1)
        → (5(2(2()())())(7(3(8()())(12()()))(18()())))
(applique-feuilles even? a1)
        → (5(2(#f()())())(7(3(#t()())(#t()()))(#f()())))
```

Abstraction

• En utilisant la fonction abstraite sur les arbres vue en cours, écrire une fonction qui retourne la liste des valeurs des nœuds d'un arbre dans l'ordre infixe.

```
(affiche a1) → (1 2 5 4 3 6 7 9)
```

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)
```

• Écrire une fonction qui ajoute 1 à chaque élément d'une liste de nombres.

```
(plus_1 '(3 1 7 4 5)) → (4 2 8 5 6)
```

• Écrire une fonction qui calcule la somme et le produit d'une liste de nombres.

```
(somme_prod '(3 1 7 4 5)) → (20 420)
```

• Écrire une fonction qui calcule la somme des valeurs absolues d'une liste de nombres.

```
(somme_abs '(3 -2 1 -4)) → 10
```

• Écrire une fonction qui compte le nombre d'éléments pairs d'une liste de nombres entiers.

```
(nb_pairs '(1 2 3 4 5)) → 2
```

• Écrire une fonction qui nie chaque élément d'une liste de listes de booléens.

```
(not_listes '((#t #f #f) (#f #t))) → ((#f #t #t) (#t #f))
```


Révisions

• Écrire une fonction qui retourne le $i^{ème}$ élément d'une liste.

• En utilisant la fonction précédente, écrire une fonction qui étant donnée une liste L de couples (i, sl) retourne la liste composée pour chaque couple de L du $i^{ème}$ élément de sl.

```
(iemecouples '((3 (a b c d)) (1 (e z)) (2 (q s d f)))) → (c e s)
```

• Écrire une fonction qui étant donnée une liste de nombres (x1 x2 … xn) calcule $\sum_{i=1}^{n} ix_i$

```
(sigma '(4 2 1 3)) → 23
```

• Écrire une fonction qui, étant donnée une liste L quelconque, construit la liste plate des nombres trouvés en profondeur dans L.

```
(listenb '(a 3 (r 52 1 (2 j) t) 5)) → (3 52 1 2 5)
```

• Écrire une fonction qui, étant donnés un nombre x et un ABR de nombres a, retourne la profondeur du nœud de valeur x dans a.

```
(prof 1 '(3(2(1()())())(7(5(4()())(6()()))(9(8()())())))) → 2
```

• Écrire une fonction qui, étant données une fonction f et une liste L de couples (booléen, liste), retourne la liste des listes de L auxquelles on a appliqué la fonction f quand le booléen est vrai.

```
(appliquecouples cdr '((#t (a z e r)) (#f (q s d)) (#t (z d v))))
→ ((z e r) (q s d) (d v))
```

• Utiliser la fonction précédente pour définir la fonction *carres* définie ci-dessous :

```
(carres '((#t (1 5 8 7)) (#f (2 4 2)) (#t (1 8 4))))
→ ((1 25 64 49) (2 4 2) (1 64 16))
```

• Définir une autre version de la fonction *iemecouples* en utilisant la fonction map.

• Définir une autre version de la fonction *appliquecouples* en utilisant la fonction map.