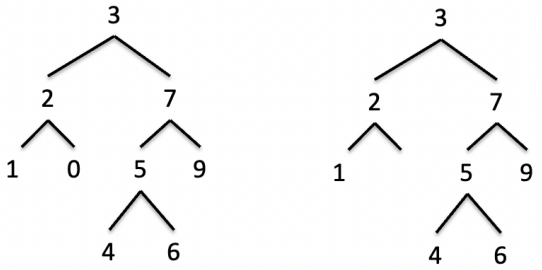
TDTP 6 : récursivité profonde, début des arbres

À préparer avant la séance

• Définir une fonction qui compte en profondeur le nombre d'éléments d'une liste.

```
(compter '(1 2 (q w e) (r (e (w t)) 3 (e)) (o))) → 12
```

• Copier les primitives sur les arbres définies en cours (via C5 ou le fichier de jeu de tests), puis définir les arbres a1 et a2 suivants :



Attention pour la primitive `arbre-vide` *qui n'a pas de paramètre, il faut passer dans le niveau de langage "étudiant avancé".*

Vérifiez avec la primitive `arbre?` que vos 2 arbres sont bien définis, puis familiarisez-vous avec les primitives en évaluant les expressions suivantes :

```
(arbre? (arbre-vide))
(arbre? a1)
(arbre? '(3 4))
(arbre? '(3 4 (5)))
(arbre? '(3 (4) 5))
(arbre? '(3 (4) (5)))
(arbre-vide)
(arbre-vide? a1)
(arbre-vide? a2)
(arbre-vide? (arbre-vide))
```

```
(valeur a1)
(fils-g a1)
(fils-d a1)
(cons-binaire 1 (arbre-vide) (arbre-vide))
(cons-binaire '* a1 a2)
(arbre=? a1 a1)
(arbre=? a1 a2)
(feuille? '(1 (arbre-vide) (arbre-vide)))
(feuille? a1)
(feuille? (arbre-vide))
```

À faire pendant la séance

Récursivité profonde

TD • Définir une fonction qui calcule en profondeur le nombre d'occurrences d'un élément dans une liste.

```
(occ-prof 'a '(a (b) (c ((a e) f)))) → 2
```

TP • Définir une fonction qui, étant donnée une liste L, ajoute 2 à chaque nombre de L en profondeur.

```
(ajoute2 '(2 b (10 a (56 3) 5) 4)) → (4 b (12 a (58 5) 7) 6)
```

TP • Définir une fonction qui, étant donnée une liste contenant des entiers, remplace en profondeur les entiers de la liste par un booléen spécifiant si l'entier correspondant est pair ou non.

```
(pairs '(2 (3) 5 (6 (7)))) → (#t (#f) #f (#t (#f)))
```

Arbres binaires

TD • Définir une fonction qui compte le nombre de nœuds d'un arbre.

```
(nb-noeuds a1) → 9
```

TD • Définir une fonction booléenne qui teste si une valeur appartient à un arbre.

```
(appart? 7 a1) → #t
```

TD • Définir une fonction qui retourne la liste résultant du parcours infixe d'un arbre : en chaque nœud, on parcourt le fils gauche, puis on note la valeur du nœud, puis on parcourt le fils droit.

```
(infixe a1) → (1 2 0 3 4 5 6 7 9)
```

TP • Définir une fonction qui retourne la liste des valeurs des feuilles d'un arbre.

```
(feuilles a1) → (1 0 4 6 9)
```

Pour s'entraîner (exercices supplémentaires facultatifs)

• Définir une fonction qui remplace en profondeur tout nombre d'une liste quelconque par sa valeur absolue (pensez à utiliser la fonction prédéfinie `abs`).

```
(absliste '(& -3 2 (1 (2 "cvb") -4) r)) → (& 3 2 (1 (2 "cvb") 4) r)
```

• Définir une fonction qui calcule la profondeur d'une liste.

```
(profondeur '(a (b c) d)) -> 2
(profondeur '(a (b) (c ((d e) f)))) -> 4
```

• Définir une fonction qui compte le nombre de feuilles d'un arbre.

```
(nb-feuilles a1) → 5
```