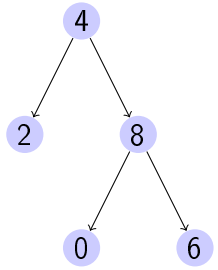
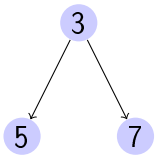
TD-TP n°6

Arbres, premières fonctions et parcours.

Télécharger sur Moodle le fichier intitulé Primitives.rkt, et copier/coller les fonctions de construction sur les arbres en haut de votre éditeur Racket.

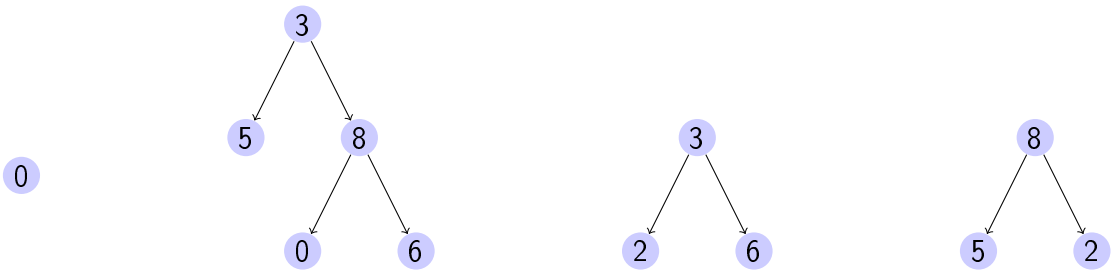
Exercice 1. Définir les arbres suivants en Racket:

$a_1 =$ 

$a_2 =$ 

1. Prédire les résultats des instructions Racket suivantes:
   - `(arbre-vide? a1)`
   - `(arbre-vide? (fils-g a2))`
   - `(feuille? (fils-d a2)`
   - `(fils-d a1)`
   - `(fils-g (fils-d a1))`
   - `(attache-arbre (racine a1) (fils-g a2) (fils-d a2))`
   - `(attache-arbre 2 (fils-d a2) (arbre-vide))`

2. Déterminer les commandes Racket, utilisant a_1 et a_2 (sans rentrer de nombre entier à la main) et permettant d'obtenir les arbres suivants:



Exercice 2.

1. Définir une fonction *adesfils* qui retourne $\#t$ si l'arbre possède un fils gauche et un fils droit non vides, faux sinon.

2. Définir une fonction *hauteur* permettant de calculer la hauteur d'un arbre, c'est-à-dire le nombre d'étages qui le compose.

   `(hauteur a1)` renvoie 3. `(hauteur a2)` renvoie 2.

3. Écrire une fonction *arbrepair*, étant donné un arbre de nombres, remplace les valeurs impaires par 0.

   `(arbrepair a2)` renvoie `'(0 (0 () ()) (0 () ()))`.

4. Définir une fonction *min-arbre* qui calcule le minimum d'un arbre binaire contenant des nombres.

   `(min-arbre a1)` renvoie `0` `(min-arbre a2)` renvoie `3`.

5. Définir une fonction *feuilles* qui retourne la liste des feuilles d'un arbre.

   `(feuilles a1)` renvoie `'(2 0 6)` `(feuilles a2)` renvoie `'(5 7)`.

6. Définir une fonction *prefixe* qui retourne la liste résultant du parcours préfixe d'un arbre.

7. Définir une fonction *remplace-hauteur* qui, étant donné un arbre, remplace la valeur de chaque nœud par sa hauteur dans l'arbre.

   `(remplace-hauteur a1)` renvoie `'(1 (2 () ()) (2 (3 () ()) (3 () ())))`.

8. Définir une fonction *remplace-feuille* qui, étant donné un arbre de nombres, remplace la valeur de chaque nœud par la somme des feuilles accessibles depuis ce nœud.

   `(remplace-feuille a1)` renvoie `'(8 (2 () ()) (6 (0 () ()) (6 () ())))`.

Exercice 3. Parcours en largeur

1. Définir une fonction *liste-racines* qui prend en paramètre **une liste d'arbres** L, et retourne la liste des nœuds racines de chacun des arbres de la liste.

   `(liste-racines '(a1 a2))` renvoie `'(4 3)`.

2. Définir une fonction *sous-arbres* qui prend en paramètre **une liste d'arbres** L, et retourne la liste de tous les sous-arbres (fils gauche et droit) des arbres de la liste.

3. Définir, en utilisant une récursivité terminale, une fonction qui prend en paramètre un arbre a, et renvoie la liste des sommets de a parcouru par un sommet en largeur.