UFR IM²AG
UNIVERSITÉ Grenoble Alpes

DIU EIL bloc 2
Année 2020 – 2021

Arbres n-aires

Objectifs

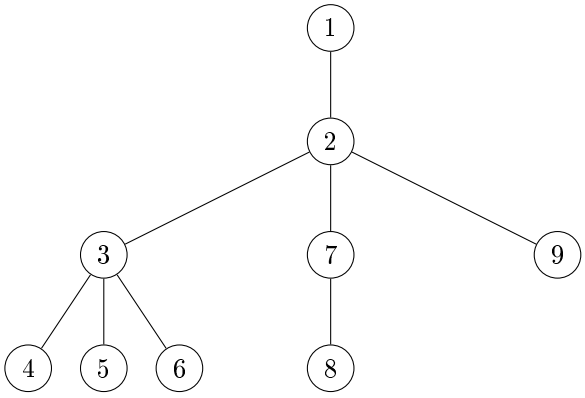
À la fin de cette séance, vous devriez être capable de :
- manipuler et concevoir des arbres n-aires comme des structures abstraites ;
- réfléchir aux propriétés des arbres n-aires ;
- proposer des implémentations d'arbres n-aires cohérentes avec les spécifications choisies en utilisant des structures sous-jacentes adaptées.

On s'intéresse ici à l'implémentation d'un type abstrait «Arbre n-aire», en utilisant le type «Arbre binaire» vu lors de la séance précédente.

Un arbre n-aire est ici implémenté comme suit :
- le «fils aîné» d'un nœud n-aire est implémenté par le «fils gauche» du nœud binaire du père ;
- le «frère cadet» d'un nœud n-aire est implémenté par le «fils droit» du nœud binaire du frère précédent.

Exercice 1.

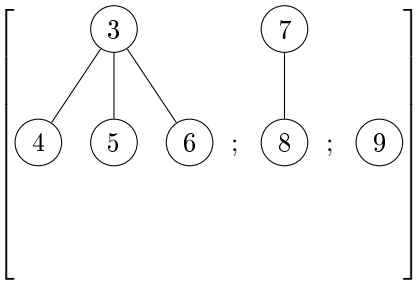
Q 1. Traduisez l'arbre ci-dessous dans sa représentation binaire :



Nous aurons également besoin d'une notion de liste d'arbres : ici nous allons **également** représenter une liste par un arbre binaire : le premier élément de la liste se trouve à la racine de l'arbre, et les éléments suivants sont situés dans son fils droit (qui est inutilisé pour l'instant, regardez votre arbre de la question précédente pour vous en convaincre).

Il peut sembler confus d'utiliser la même structure sous-jacente (les arbres binaires) pour implémenter deux types différents, mais il faut se souvenir qu'on utilisera des primitives de manipulation différentes pour les listes et pour les arbres n-aires, il n'y aura donc pas d'ambiguïté.

Q 2. Traduisez la liste d'arbres ci-dessous dans sa représentation binaire :



Exercice 2.

Q 2. Implémentez les primitives du type abstrait «arbre n-aire», telles que définies ci-dessous.

Élément : un **type**
Arbre : un **type**
ListeArbre : un **type**

ArbreVide
{ **Données :** *aucun*
Résultat : *un Arbre vide*}

NouveauNœud
{ **Données :** *un Élément x, une ListeArbre L*
Résultat : *un Arbre constitué du nœud x, dont les fils sont les éléments de L*
Effet de bord : *un nouveau nœud a été créé* }

EstArbreVide
{ **Données :** *un Arbre A*
Résultat : *un booléen vrai ssi A est un Arbre vide* }

Elem
{ **Données :** *un Arbre A*
Résultat : *l'Élément associé àla racine de A*
Pré-condition : *A est non vide* }

ListeFils
{ **Données :** *un Arbre A*
Résultat : *une ListeArbre*
description : *A doit être non vide, renvoie la liste des fils associée àla racine de A* }

{ *Manipulation des listes d'arbres* }

ListeVide
{ **Données :** *aucun*
Résultat : *une ListeArbre vide* }

Cons
{ **Données :** *un Arbre A, une ListeArbre L*
Résultat : *une ListeArbre constituée de l'arbre A, suivi de la liste L.* }

EstListeVide
{ **Données :** *une ListeArbre L*
Résultat : *un booléen vrai ssi L est vide.* }

Premier
{ **Données :** *une ListeArbre L*
Résultat : *un Arbre, renvoie le premier arbre de la liste L*
Pré-condition : *L non vide* }

Suivants
{ **Données :** *une ListeArbre L*
Résultat : *une ListeArbre, renvoie la liste des arbres suivants le premier.*
Pré-condition : *L non vide* }