Hugo CASTELL - Arthur LEVI
1A FISA SN

Raffinage Arbre Généalogique :

Init ✅
R0 : **Créer un arbre minimal contenant le seul nœud racine, sans père ni mère.**
    Binary_Tree(GenealogicTree, Root)

AddParent ✅
R0 : **Ajouter un parent** à un noeud donné.
    Si c'est une femme Alors
        Node.Mother ← MotherNode
    Sinon
        Node.Father ← FatherNode
    FinSi

NumberAncestors ✅
R0 : **Obtenir le nombre d'ancêtres connus** (lui compris) d'un individu donné.
Exemple : Sur l'arbre ci-dessus, le nombre d'ancêtres connus de 2 est 5.
R1 : Comment "Obtenir le nombre d'ancêtres connus (lui compris) d'un individu donné" ?
    Créer un compteur en l'initialisant à 1 car on compte l'individu donné
    Parcourir tous les ancêtres de l'individu et incrémenter le compteur pour chaque ancêtre connu
R2 : Comment "Créer un compteur en l'initialisant à 1 car on compte l'individu donné" ?
    ancestor_count ← 1
R2 : Comment "Parcourir tous les ancêtres de l'individu et incrémenter le compteur pour chaque ancêtre connu" ?
    Chercher les ancêtres directs connus et les compter
    Récursivement, pour chaque ancêtre direct connu, répéter la recherche
R3 : Comment "Chercher les ancêtres directs connus et les compter" en le faisant récursivement ?
    Si père connu Alors
        ancestor_count ← NumberAncestors(Pere) + 1
    FinSi
    Si mère connue Alors
        ancestor_count ← NumberAncestors(Mere) + 1

FinSi

AncestorsGen ✅
R0 : **Obtenir l'ensemble des ancêtres situés à une certaine génération** d'un nœud donné.
Exemple, les ancêtres de génération 2 du nœud 18 sont les nœuds 15, 26 et 33. Les ancêtres de génération 1 du nœud 33 sont les nœuds 25 et 42.
R1 : Comment "Obtenir l'ensemble des ancêtres situés à une certaine génération" ?
Créer deux listes current_gen et next_gen, en initialisant current_gen avec l'identifiant du nœud donné.
Parcourir old_gen jusqu'à la génération donnée
R2 : Comment "Créer deux listes current_gen et next_gen, en initialisant current_gen avec l'identifiant du nœud donné." ?
current_gen ← [root]
next_gen ← []
current_gen_number ← 1
R2 : Comment "Parcourir old_gen jusqu'à la génération donnée" ?
TantQue current_gen_number < gen_attendue Faire
Parcourir les individus de la current_gen et écraser la génération courante avec tous les ancêtres directs
FinTantQue
R3 : Comment "Parcourir les individus de la current_gen et écraser la génération courante avec tous les ancêtres directs" ?
Pour chaque individu dans current_gen Faire
Si père connu Alors
Add(father, next_gen)
FinSi
Si mère connue Alors
Add(mother, next_gen)
FinSi
FinPour

current_gen ← next_gen
next_gen ← []

Hugo CASTELL - Arthur LEVI
1A FISA SN

PrintTree ✅

R0 : **Afficher l'arbre** à partir d'un nœud donné

R1 : Comment "Afficher l'arbre à partir d'un nœud donné" ?

    Parcourir l'arbre et trouver le noeud donné dans l'arbre
    Afficher les ancêtres du noeud

R2 : Comment "Parcourir l'arbre et trouver le noeud donné dans l'arbre" ?

```
searched_node ← null
current_gen ← [root]
next_gen ← []

TantQue searched_node = null and current_gen is not empty Faire
        Pour chaque individu dans current_gen Faire
                Si identifiant noeud cherché = identifiant noeud courant Alors
                        searched_node  ← current_node
                Sinon
                        Si père connu Alors
                                Add(father, next_gen)
                        FinSi
                        Si mère connue Alors
                                Add(mother, next_gen)
                        FinSi
                FinSi
        FinPour

        current_gen ← next_gen
        next_gen ← []
FinTantQue
```

R2 : Comment "Afficher les ancêtres du noeud" ?

    Parcourir récursivement les ancêtres du noeud
    Afficher chaque ancêtre

R3 : Comment “Parcourir récursivement les ancêtres du noeud” ?

```
Print(searched_node, 0)
procedure Print(Node, numero_gen) :
        Afficher le noeud
        Si père connu Alors
                Print(father, numero_gen + 1)
        FinSi
        Si mère connue Alors
                Print(mother, numero_gen + 1)
        FinSi
```

R3 : Comment “Afficher chaque ancêtre” ?

```
Décaler selon la génération
Afficher le noeud
```

R4 : Comment “Décaler selon la génération” ?

```
Put(“\n”)
Pour i allant de 0 à num_gen Faire
        Put(“\t”)
FinPour
```

R4 : Comment “Afficher le noeud” ?

```
Put(node.identifiant)
```

Hugo CASTELL - Arthur LEVI
1A FISA SN

DeletePerson ✅

R0 : **Supprimer, pour un arbre, un nœud et ses ancêtres**.

Exemple, si on veut supprimer le nœud 15 et ses ancêtres, on supprime le nœud 5.

R1 : Comment “Supprimer, pour un arbre, un nœud et ses ancêtres” ?

Parcourir l’arbre et trouver le noeud à supprimer dans l’arbre
Supprimer les ancêtres du noeud et le noeud

R2 : Comment “Parcourir l’arbre et trouver le nœud à supprimer dans l’arbre” ?

```
searched_node ← null
current_gen ← [root]
next_gen ← []

TantQue searched_node = null and current_gen is not empty Faire
        Pour chaque individu dans current_gen Faire
                Si identifiant noeud cherché = identifiant noeud courant Alors
                        searched_node  ← current_node
                Sinon
                        Si père connu Alors
                                Add(father, next_gen)
                        FinSi
                        Si mère connue Alors
                                Add(mother, next_gen)
                        FinSi
                FinSi
        FinPour

        current_gen ← next_gen
        next_gen ← []
FinTantQue
```

R2 : Comment “Supprimer les ancêtres du nœud et le nœud” ?

Parcourir récursivement les ancêtres du noeud

Supprimer chaque ancêtre (à l'envers, en partant des feuilles)

R3 : Comment "Parcourir récursivement les ancêtres du nœud" ?

Delete(searched_node)

procedure Delete(node) :

Si père connu Alors

Delete(father)

FinSi

Si mère connue Alors

Delete(mother)

FinSi

Supprimer chaque ancêtre

R3 : Comment "Supprimer chaque ancêtre (à l'envers, en partant des feuilles)" ?

node ← null

Hugo CASTELL - Arthur LEVI
1A FISA SN

PersonsWithXParents ✅

R0 : Obtenir l'ensemble des **individus qui n'ont X parents connus** (x appartenant à [0..2])

R1: Comment "Obtenir l'ensemble des individus qui n'ont X parents connus" ?

    Créer une liste des noeuds ayant x parents dans l'arbre
    Parcourir l'ensemble des noeuds de l'arbre pour les trouver

R2 : Comment "Créer une liste des noeuds ayant x parents dans l'arbre" ?

    corresponding_nodes ← []

R2 : Comment "Parcourir l'ensemble des noeuds de l'arbre pour les trouver" ?

    current_gen ← [root]
    next_gen ← []

    TantQue searched_node = null and current_gen is not empty Faire
        Pour chaque individu dans current_gen Faire
            parents_count ← 0
            Vérifier s'ils ont x parents et ajouter les parents à la boucle
            Si parents_count = x Alors
                Les stocker dans la liste
            FinSi
        FinPour

        current_gen ← next_gen
        next_gen ← []
    FinTantQue

R3 : Comment "Vérifier s'ils ont x parents et ajouter les parents à la boucle" ?

    Si père connu Alors
        Add(father, next_gen)
        parents_count ← parents_count + 1
    FinSi
    Si mère connue Alors
        Add(mother, next_gen)

```
            parents_count ← parents_count + 1
        FinSi
R3 : Comment “Les stocker dans la liste” ?
        Add(node, corresponding_nodes)

# FindPersonInTree
R0 : Savoir si un individu est dans l’arbre.
R1 : Comment “Savoir si un individu est dans l’arbre” ?
        Parcourir l’arbre et trouver le noeud à supprimer dans l’arbre
R2 : Comment “Parcourir l’arbre et trouver le noeud à supprimer dans l’arbre” ?
        searched_node ← null
        current_gen ← [root]
        next_gen ← []

        TantQue searched_node = null and current_gen is not empty Faire
            Pour chaque individu dans current_gen Faire
                Si identifiant noeud cherché = identifiant noeud courant Alors
                    searched_node  ← current_node
                Sinon
                    Si père connu Alors
                        Add(father, next_gen)
                    FinSi
                    Si mère connue Alors
                        Add(mother, next_gen)
                    FinSi
                FinSi
            FinPour

            current_gen ← next_gen
            next_gen ← []
        FinTantQue
```

Hugo CASTELL - Arthur LEVI
1A FISA SN