Master BioInformatique	ANNÉE : 2008/2009 SESSION DE JANVIER 2009 **PARCOURS :** Master 1 **UE :** Algorithmes et structures de données **Épreuve :** Examen **Date :** Jeudi 18 décembre 2008 **Heure :** 13 heures 30 **Durée :** 2 heures Documents : autorisés Épreuve de M. Alain GRIFFAULT

Code d'anonymat :

Avertissement

- Les deux premiers exercices sont similaires mais indépendants. Il est conseillé de commencer par le premier. Le troisième dépend des deux premiers. Les questions marquées d'un astérisque sont plus difficiles.
- Le barème total est de 24 points car le sujet est assez long.
- L'espace laissé pour les réponses est suffisant (sauf si vous utilisez ces feuilles comme brouillon, ce qui est fortement déconseillé).

Contexte

Une personne souhaite informatiser les arbres généalogiques de sa famille. Ses notions informatique sont suffisantes pour qu'il aperçoive un lien entre la structure d'arbre qu'il a apprise, et les dessins retrouvés chez lui.
Après mûre réflexion, il pense que pour l'arbre des descendants, il devra utiliser un arbre avec une liste de pointeurs pour pouvoir mettre les enfants ; alors que du fait que la liste des parents est toujours réduite à un père et une mère, il pense qu'un arbre binaire est la bonne solution pour l'arbre des ascendants.

Exercice 1 (Arbres généalogiques des ascendants (10 points))
Il décide donc de commencer par l'arbre des ascendants, et utilise la structure suivante pour stocker l'information sur les personnes.

```
structure Enfant {
  nom, prenom : String;
  neLe : Date;
  pere, mere : *Enfant;
}

structure Ascendant {
  asc : *Enfant;
}
```

Question 1.1 (0,5 point) *Dessiner la structure obtenue après les créations suivantes d'enfants. Le prénom et l'âge des personnes sont omis pour simplification.*

1. *creerEnfant(mamie1, pere :nil, mere :nil)*
2. *creerEnfant(mamie2, pere :nil, mere :nil)*
3. *creerEnfant(papy, pere :nil, mere :nil)*
4. *creerEnfant(papa, pere :papy, mere :mamie1)*
5. *creerEnfant(maman, pere :papy, mere :mamie2)*
6. *creerEnfant(moi, pere :papa, mere :maman)*
7. *creerAscendant(moi)*

Question 1.2 *0,5 point] La structure obtenue est elle toujours un arbre ? Justifier.*

Question 1.3 (1 point) *Démontrer (en utilisant un argument simple) que la structure obtenue dans le cas d'un arbre généalogique des ascendants ne contient pas de cycle.*

Convaincu que cette structure est proche d'un arbre binaire, la personne décide d'utiliser les algorithmes classiques de parcours d'arbres binaires.

```
Parcours_Recursif_Arbre_Ascendants(asc) {
    si (asc <> nil) alors {
        Traitement_Noeud_Prefixe(asc);
        Parcours_Recursif_Arbre_Ascendant(asc.pere);
        Traitement_Noeud_Infixe(asc);
        Parcours_Recursif_Arbre_Ascendant(asc.mere);
        Traitement_Noeud_Postfixe(asc);
    }
}
```

Question 1.4 (1 point) *Donner la sortie de l'algorithme précédent dans le cas d'un affichage infixe de la structure obtenue à la question 1.1. Expliquer le résultat.*

Considérant l'anomalie précédente comme non grave, car facilement réparable, la personne décide d'adapter les algorithmes sur les arbres binaires à ses arbres ascendants.

Question 1.5 (3 points) *Donner un algorithme* `Parcours_Recursif_Arbre_Ascendants_Niveau(asc, n)` *qui affiche uniquement la liste des ascendants de niveau n d'une personne. Le niveau 0 est la personne elle même, le niveau 1 ses parents, le niveau 2 ses grands-parents et ainsi de suite. Les doublons d'affichage seront autorisés, et l'affichage sera en ordre infixe.*

Question 1.6 (1 point) *Pour une personne donnée, est-il possible qu'une même autre personne soit dans plusieurs listes de niveaux différents ? Si oui, dessiner une structure correspondante ; si non, expliquer pourquoi.*

Question 1.7 (3 points (*)) *Donner un algorithme* `Parcours_Recursif_Arbre_Ascendants_Comptage(asc)` *qui compte les ascendants d'une personne. On pourra ajouter dans la structure* `Enfant` *un booléen* `visite` *qui indique si une cellule à déjà été visitée lors de ce parcours. On supposera que cette variable est correctement initialisée.*

Exercice 2 (Arbres généalogiques des descendants (9 points))
La personne décide d'implémenter les arbres descendants en utilisant la structure d'arbre. Les enfants seront mis dans une liste de pointeurs. Il utilise donc la structure suivante pour stocker l'information sur les personnes.

```
structure Parent {
  nom, prenom : String;
  neLe : Date;
  enfants : *Liste_Freres_Soeurs;
}
structure Liste_Freres_Soeurs {
  qui : *Parent;
  suivant : *Liste_Freres_Soeurs;
}
structure Descendant {
  desc : *Parent;
}
```

Question 2.1 (0,5 point) *Dessiner la structure obtenue après les créations suivantes de parents. Le prénom et l'âge des personnes sont omis pour simplification.*

1. *creerParent(un, enfants : nil)*
2. *creerParent(deux, enfants : nil)*
3. *creerParent(trois, enfants : nil)*
4. *creerParent(quatre, enfants : nil)*
5. *creerParent(pere1, enfants : un, trois, nil)*
6. *creerParent(pere2, enfants : quatre, nil)*
7. *creerParent(mere1, enfants : un, deux, nil)*
8. *creerParent(mere2, enfants : trois, quatre, nil)*
9. *creerParent(moi, enfants : pere1, pere2, mere1, mere2, deux, nil)*
10. *creerDescendant(moi)*

Question 2.2 (0,5 point) *La structure obtenue est elle toujours un arbre ? Justifier.*

Question 2.3 (1 point) *Démontrer (en utilisant un argument simple) que la structure obtenue dans le cas d'un arbre généalogique des descendants ne contient pas de cycle.*

Convaincu que cette structure est proche d'un arbre, la personne décide d'utiliser les algorithmes classiques de parcours d'arbres.

```
Parcours_Recursif_Arbre_Descendants(desc) {
    si (desc <> nil) alors {
        Traitement_Noeud_Prefixe(desc);
        Aux <- desc.enfants;
        tant que (Aux <> nil) faire {
            Parcours_Recursif_Arbre_Descendants(Aux.qui);
            Aux <- Aux.suivant;
        }
        Traitement_Noeud_Postfixe(desc);
    }
}
```

Question 2.4 (1 point) *Donner la sortie de l'algorithme précédent dans le cas d'un affichage préfixe de la structure obtenue à la question 2.1. Expliquer le résultat.*

Considérant l'anomalie précédente comme non grave, car facilement réparable, la personne décide d'adapter les algorithmes sur les arbres à ses arbres descendants.

Question 2.5 (3 points) *Donner un algorithme* `Parcours_Recursif_Arbre_Descendants_Niveau(desc, n)` *qui affiche uniquement la liste des descendants de niveau n d'une personne. Le niveau 0 est la personne elle même, le niveau 1 ses enfants, le niveau 2 ses petits-enfants et ainsi de suite. Les doublons d'affichage seront autorisés et l'affichage sera en ordre préfixe.*

Question 2.6 (3 points (*)) *Donner un algorithme* `Parcours_Recursif_Arbre_Descendants_Comptage(desc)` *qui compte les descendants d'une personne. On pourra ajouter dans la structure* `Parent` *un booléen* `visite` *qui indique si une cellule à déjà été visitée lors de ce parcours. On supposera que cette variable est correctement initialisée.*

Exercice 3 (Arbres généalogiques (5 points))
De plus en plus fier d'elle même, la personne décide d'intégrer dans une même structure les deux précédentes afin d'avoir les ascendants et les descendants réunis.

```
structure Personne {
  nom, prenom : String;
  neLe : Date;
  pere, mere : *Personne;
  enfants : *Liste_Freres_Soeurs;
}

structure Liste_Freres_Soeurs {
  qui : Personne;
  suivant : *Liste_Freres_Soeurs;
}

structure Genealogie {
  pers : *Personne;
}
```

Vous ferez l'hypothèse que les algorithmes des deux premiers exercices sont utilisables sans modification avec cette nouvelle structure.

Question 3.1 (2 points) *Donner un algorithme* `Freres_Soeurs_Arbre_Genealogique(pers)` *qui affiche pour une personne : la personne, ses frères, ses demi-frères, ses soeurs et ses demi-soeurs. Les doublons d'affichage sont acceptés.*

Question 3.2 (3 points (*)) *Donner un algorithme* `Cousins_Distants_Arbre_Genealogique(pers, n)` *qui affiche les cousins et cousines de distance n. La distance 0 correspond à la personne elle-même, la distance 1 à l'algorithme précédent, la distance 2 correspond à l'ensemble des cousins (enfants des parents, des oncles et des tantes), la distance 3 correspond aux cousins issus de germain (petits-enfants des grands-parents, des grands-oncles et des grandes-tantes), et ainsi de suite. Les doublons d'affichage sont acceptés.*