Contrôle continu 3
Pointeurs - Récursivité – Listes – Piles – Files– Arbres
(3 pages)

Questions de cours (5 points)

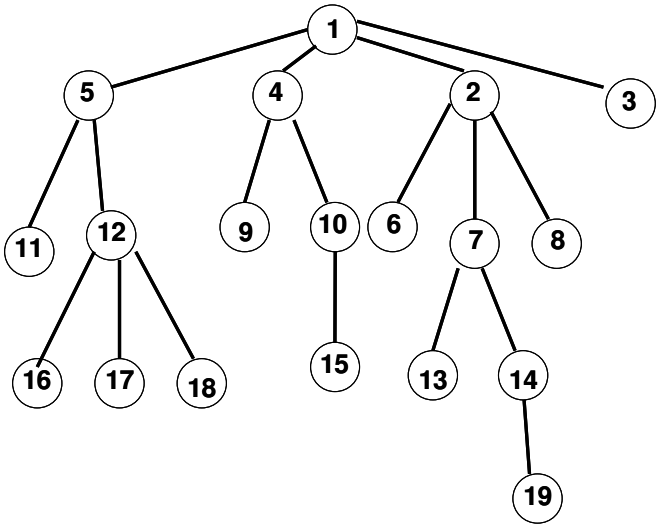
- Peut-on créer un arbre binaire dont la valeur du sommet est une pile d'entiers?
- Quelle structure de données est la plus efficace pour parcourir un arbre en ordre hiérarchique ?
- Quel parcours existe pour les arbres binaires et n'existe pas pour les arbres planaires ?
- Dessinez la mémoire, après la suite d'opérations suivante :

```
var A : arbreBinaire de listeSC d'entier;
var s :listeSC d'entier;
creerListe(s) ;
insérerEnTete(s,0);
insérerEnTete(s,1);
A=creerArbreBinaire(s) ;
creerListe(s) ;
insérerEntete(s,2) ;
débutListe(s) ;
ajouterFilsGauche(A,s)
insérerApres(s,3) ;
suivant(s) ;
ajouterFilsDroit(A,s)
```

Que remarquez-vous qui peut causer des erreurs?

Exercice 2 (5 points)

- Soit l'arbre planaire suivant



1. Donnez sa hauteur
2. Donnez la liste des sommets dans l'ordre préfixe
3. Donnez la liste des sommets dans l'ordre hiérarchique
4. Donnez l'arbre binaire correspondant dans la bijection de Knuth
5. Pour l'arbre binaire ainsi obtenu, donner la liste des entiers en ordre préfixe, en ordre infixe et en ordre suffixe.

Exercice 2 (10 points)

Ecrire une fonction qui renvoie la liste de toutes les feuilles d'un arbre binaire :
1 – En utilisant les primitives du type arbreBinaire et listeSC
2 – En implémentant directement en allocation dynamique pour le type Arbre Binaire et le type listeSC.

Listes simplement chainées (listeSC)

```
fonction valeur(val L:liste d'objet):objet;
fonction debutListe(val L:liste d'objet);
fonction suivant(val L:liste d'objet);
fonction listeVide(val L:liste d'objet): booleen;
fonction créerListe(ref L:liste d'objet):vide;
fonction insérerAprès(ref L:liste d'objet; val x:objet;):vide;
fonction insérerEnTete(ref L:liste d'objet  val x:objet):vide;
fonction supprimerAprès(ref L:liste d'objet):vide;
fonction supprimerEnTete(ref L:liste d'objet):vide;
```

Listes doublement chainées (listeDC)

```
fonction finListe(val L:liste d'objet):vide;
fonction précédent(val L::liste d'objet): vide;
```

Piles

```
fonction valeur(ref P:pile de objet):objet;
fonction fileVide(ref P:pile de objet):booléen;
fonction créerPile(P:pile de objet);
fonction empiler(ref P:pile de objet;val x:objet):vide;
fonction dépiler(ref P:pile de objet):vide;
fonction detruirePile(ref P:pile de objet):vide;
```

Files

```
fonction valeur(ref F:file de objet):objet;
fonction fileVide(ref F:file de objet):booléen;
fonction créerFile(F:file de objet);vide;
fonction enfiler(ref F:file de objet;val x:objet):vide;
fonction défiler (ref F:file de objet):vide;
fonction detruireFile(ref F:file de objet):vide;
```

Arbres binaires

```
fonction getValeur(val S:sommet):objet;
fonction filsGauche(val S:sommet):sommet;
fonction filsDroit(val S:sommet):sommet;
fonction pere(val S:sommet):sommet;
fonction setValeur(ref S:sommet;val x:objet):vide;
fonction ajouterFilsGauche(ref S:sommet,val x:objet):vide;
fonction ajouterFilsDroit(ref S:sommet,x:objet):vide;
fonction supprimerFilsGauche(ref S:sommet):vide;
fonction supprimerFilsDroit(ref S:sommet):vide;
fonction detruireSommet(ref S:sommet):vide;
fonction créerArbreBinaire(val Racine:objet):sommet;
```

Arbres planaires

```
fonction valeur(val S:sommetArbrePlanaire):objet;
fonction premierFils(val S:sommetArbrePlanaire):sommetArbrePlanaire;
fonction frere(val S:sommetArbrePlanaire):sommetArbrePlanaire;
fonction pere(val S:sommetArbrePlanaire):sommetArbrePlanaire;
fonction créerArborescence(val racine:objet):sommetArbrePlanaire;
fonction ajouterFils(ref S:sommetArbrePlanaire,val x:objet):vide;
fonction supprimerSommet(ref S:sommetArbrePlanaire):vide;
fonction créerArbreBPlaniare(val Racine:objet):sommet;
```