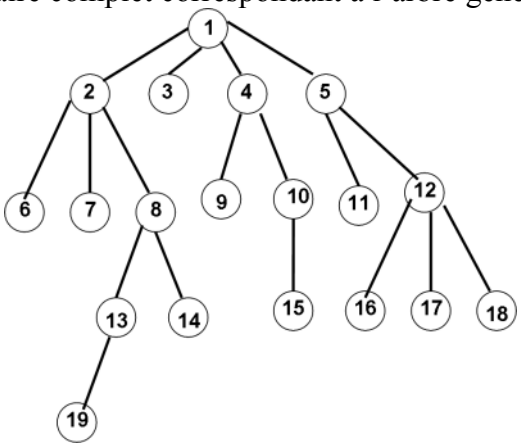
Contrôle continu 2007-2008
INF251
Pointeurs - Récursivité – Listes – Piles – Files– Arbres

Questions de cours (5 points)

- Donnez la structure d'un sommet pour un arbre binaire en allocation dynamique.
- Un sommet d'un arbre peut-il contenir une chaine de caractère ?
- Donnez l'arbre binaire complet correspondant à l'arbre général suivant

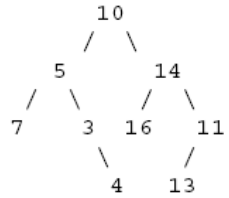


- Dessinez la mémoire, après la suite d'opérations suivante :

```
var A : arbreBinaire de ^entier;
var s :^entire;
var p:^sommet;
new(s) ;
s^=0;
creerArbre(A,s) ;
p=racine(A);
pour i=1 à 3 pas 2 faire
  new(s) ;
  s^=2*i;
  ajouterFilsGauche(P,s)
  new(s) ;
  s^=2*i+1;
  ajouterFilsDroit(p,s)
   s=p;
 finpour
```

Exercice 2 (5 points)

Soit l'arbre suivant :



1 – Donner la liste des entiers en ordre préfixe, en ordre infixe et en ordre suffixe.
2 – Quelle est la hauteur de cet arbre ?
3 – Donnez un tas-min contenant les mêmes entiers
4 – Donnez un tas-max contenant les mêmes entiers

Exercice 2 (10 points)
Ecrire une fonction qui renvoie la liste des sommets d'un des chemins de la racine à une feuille de longueur hauteur(A)
1 – En utilisant les primitives du type arbreBinaire
2 – En implémentant directement en allocation dynamique.

Listes simplement chainées (listeSC)
fonction premier(val L:type_liste):^type_predefini;
fonction suivant(val L:type_liste; val P:^type_predefini):^type_predefini;
fonction listeVide(val L:type_liste):booléen;
fonction créer liste(ref L:type_liste):vide;
fonction insérerAprès(val x:type_prédéfini;ref L:type_liste; val P:^type_predefini):vide;
fonction insérerEnTete(val x:type_prédéfini;ref L:type_liste):vide;
fonction supprimerAprès(ref L:type_liste;val P:^type_predefini):vide;
fonction supprimerEnTete(ref L:type_liste):vide;
Listes doublement chainées (listeDC), On ajoute les primitives suivantes
fonction dernier(val L:type_liste):^type_predefini;
fonction précédent(val L:type_liste; val P:^type_predefini):^type_predefini;
Piles
fonction valeur(ref P:pile de type_predefini):type_predefini;
fonction pileVide(ref P:pile de type_predefini):booléen;
fonction empiler(ref P:pile de type_predefini; val v:type_predefini):vide;
fonction dépiler (ref P:pile de type_predefini):vide;
fonction créerPile(P:pile de type_predefini);
Files
fonction valeur(ref F:file de type_predefini):type_predefini;
fonction fileVide(ref F:file de type_predefini):booléen;
fonction enfiler(ref F:file de type_predefini; val v:type_predefini):vide;
fonction défiler (ref F:file de type_predefini):vide;
fonction créerFile(F:file de type_predefini);
ArbresBinaire
fonction valeurSommet(val A:arbreBinaire de type_prédéfini, val S:sommet):valeur_prédéfini;
fonction racine(val A:arbreBinaire de type_prédéfini):sommet;
fonction filsGauche(val A:arbreBinaire de type_prédéfini, val S:sommet):sommet;
fonction filsDroit(val A:arbreBinaire de type_prédéfini, val S:sommet):sommet;

fonction père(val A:arbreBinaire de type_prédéfini, val S:sommet):sommet;
fonction créerArbreBinaire(ref A:arbreBinaire de type_prédéfini, val racine:type_prédéfini):vide;
fonction ajouterFilsGauche(val x:type_prédéfini, ref A:arbreBinaire de type_prédéfini,
val S:sommet):vide;
fonction ajouterFilsDroit(val x:type_prédéfini, ref A:arbreBinaire de type_prédéfini,
val S:sommet):vide;
fonction supprimerFilsGauche(ref A:arbreBinaire de type_prédéfini, val S:sommet):vide;
fonction supprimerFilsDroit(ref A:arbreBinaire de type_prédéfini, val S:sommet):vide;

Arbres Binaires en allocation dynamique

fonction valeurSommet(val S:^sommet):valeur_prédéfini;
fonction racine(val A:arbreBinaire de type_prédéfini):^sommet;
fonction filsGauche(val S:^sommet):^sommet;
fonction filsDroit(val S:^sommet):^sommet;
fonction créerArbreBinaire(val racine:type_prédéfini):^sommet;
fonction ajouterFilsGauche(val x:type_prédéfini, val S:^sommet):vide;
fonction ajouterFilsDroit(val x:type_prédéfini, val S:^sommet):vide;
fonction supprimerFilsGauche(val S:^sommet):vide;
fonction supprimerFilsDroit(val S:^sommet):vide;
fonction grefferFilsDroit(ref s1:^sommet;ref s2:arbreBinaire):vide;
fonction grefferFilsGauche(ref s1:^sommet;ref s2:arbreBinaire):vide;
fonction elaguerFilsDroit(ref s1:^sommet):arbreBinaire;
fonction elaguerFilsGauche(ref s1:^sommet):arbreBinaire ;