INF251 Algo 1
Devoir surveillé n° 4
1h
Aucun document autorisé
09/12/2009

Question 1. Cochez les affirmations qui sont correctes : Dans tous les cas justifiez votre réponse

☐ Le temps d'accès au dernier élément d'une liste simplement chainée est en O(n).
☐ Dans un arbre binaire de recherche le minimum est toujours à la racine de l'arbre.
☐ Le temps d'accès à l'élément maximum d'un tas min est en O(1).
☐ Un arbre AVL est un tas.
☐ Pour un arbre binaire de recherche donné l'obtention de la liste des nombres triés est en temps O(n).
☐ Dans un tas, la primitive `supprimerValeur` consiste à supprimer une valeur située dans une feuille.
☐ La complexité mémoire d'une table de hachage chainée est plus importante que la complexité mémoire d'une table de hachage ouvert

Question 2.

Décrire en détail le principe d'une fonction *corrige* qui modifie la dernière lettre d'un mot du dictionnaire. On supposera que le mot se trouve dans le dictionnaire. Cette fonction prendra en argument un dictionnaire, le mot correct ainsi que la lettre à changer.

Question 3.
Un dispositif dispose de deux robots pour traiter des commandes. Une commande est identifiée par un nom sur 30 caractères et une date sous la forme [jour,mois,année]. Le premier robot (R1) mémorise les commandes dès quelles lui parviennent. Le second robot (R2) traite les commandes.

1 – Quelle structure de données pertinente peut-on utiliser pour accéder en temps constant aux commandes ?
2 – Quelle structure de données peut utiliser R1 pour que R2 traite les commandes dans le même ordre où elles arrivent à R1 ?
3 – Décrivez ces structures de données en détaillant leurs éléments.

Question 4.

Soit la suite de clé 5,8,2,6,3,4,1,7. Construire le tas Min correspondant à l'insertion consécutive des clés, on dessinera l'arbre après chacune des quatre premières insertions ainsi que l'arbre final. Montrez l'exécution de `supprimerValeur` sur le tas ainsi construit.

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) :vide
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;

Arbres binaire de recherche
fonction ajouter(ref A:arbreBinaire d'objets, val v:objet):vide;
fonction supprimer(val A: arbreBinaire d'objets, val v:objet):vide;
Tas
fonction valeur(ref T:tas d'objet): objet;
fonction ajouter(ref T:tas de objet, val v:objet):vide;
fonction supprimer(val T:tas de objet):vide;
fonction creerTas(ref T:tas,val:v:objet):vide;
fonction detruireTas(ref T:tas):vide;
Dictionnaire
fonction appartient((ref d:dictionnaire,val M::mot):booléen;
fonction creerDictionnaire(ref d: dictionnaire):vide ;
fonction ajouter(ref d:dictionnaire,val M::mot):vide;
fonction supprimer(ref d:dictionnaire,val M:mot):vide;
fonction detruireDictionnaire(ref d:dictionnaire):vide;
Table de Hachage
fonction chercher(ref T:tableHash de clés, val v:clé):curseur;
fonction créerTableHachage(ref T: tableHash de clé, ref h:fonction):vide;
fonction ajouter(ref T:tableHash de clé,val x:clé):booleen;
fonction supprimer((ref T:tableHash de clé,val x:clé):vide;