| | | |
|---|---|---|
| DISVE<br>Pôle Licence | **ANNEE UNIVERSITAIRE 2009/2010**<br>**SESSION 1 D'AUTOMNE**<br><br>**PARCOURS / ETAPE :** CSB3, MHT53 **Code UE :** INF 251<br>**Epreuve :** Algorithmes et Structures de Données Fondamentaux<br>**Date :** 21 Décembre **Heure :** 8H30 **Durée :** 1H30<br>Documents : non autorisés<br>Epreuve de M/Mme : DELEST<br>**Vous devez répondre directement sur le sujet qui comporte 8 pages.**<br>**Insérez ensuite votre réponse dans une copie d'examen comportant tous les renseignements administratifs** | |

Indiquez votre code **d'anonymat** : N° : .

**La notation tiendra compte de la clarté de l'écriture des réponses.**

Barème indicatif

- Question 1 – Connaissances générales : 2 points
- Question 2 – Containeur liés aux arbres binaires de recherche : 6 points
- Question 3 – Connaissance des structures de données : 6 Points
- Question 4 – Utilisation des structures de données : 6 points

**Question 1.** Cochez les affirmations correctes et quelle que soit la réponse justifiez sur la ligne en pointillé.

☐ Dans un arbre binaire de recherche le minimum est toujours à la racine.

……

☐ Un arbre AVL de taille 15 n'est jamais un tas.

……

☐ La structure de B-arbre permet de diminuer le temps d'accès à un élément oar rapport à un AVL.

……

☐ Si s est un pointeur vers une structure dont un des champs est n : ^entier, on accède à l'entier par s^.n^ ?

……

**Question 2.** Soit la liste de clé A=(5,4,2,7,6,8,3,1).

1 – Ecrire la fonction *ajouter* qui insère un élément dans un arbre binaire de recherche.

2 - Construire l'arbre binaire de recherche B correspondant à l'insertion consécutive des clés de la liste A. On dessinera l'arbre après chacune des quatre premières insertions ainsi que l'arbre final.

3 – Donnez pour l'arbre B la liste des sommets en ordre préfixe, en ordre infixe et en ordre suffixe.

4 - Montrez l'exécution de la suppression de la clé 4 sur l'arbre B.

5 – Construire l'AVL correspondant à l'insertion consécutive de la liste A. On dessinera l'arbre après chacune des quatre premières insertions ainsi que l'arbre final.

6 – Construire le tas Min correspondant à l'insertion consécutive de la liste A. On dessinera l'arbre final.

**Question 3.** Soit une suite de clé dans un tableau T. On considère un tableau de dimension N contenant des entiers tous différents appartenant à l'intervalle [1..N]. Par exemple, le tableau de dimension 8 est donné par T[1]=5, T[2]=8, T[3]=2, T[4]=6, T[5]=3, T[6]=4, T[7]=1, T[8]=7.

1 – Ecrire la fonction *verifier* qui vérifie que les éléments du tableau appartiennent à l'intervalle [1..N] et sont tous différents.

2 **–** En utilisant les primitives du type abstrait listeSC, écrire la fonction *tableauListe* qui a pour paramètre un tableau d'entier T et fournit en sortie une liste simplement chainée L dans le même ordre que le tableau d'entrée (sur l'exemple, L= (5,8,2,6,3,4,1,7)).

3 - Ecrire cette même fonction *tableauListe* en utilisant l'implémentation dynamique à la place des primitives

4 **–** Ecrire une fonction *creuxDeListe* qui à partir de la liste L simplement chainée fournit la file des valeurs T[j] telles que T[j-1]>T[j] et T[j]<T[j+1] (sur l'exemple [2,3,1[ où 2 est la tête de file).

5 – Ecrire cette même fonction *creuxDeListe* en utilisant l'implémentation dynamique à la place des primitives. On décrira l'implémentation choisie pour la file.

6 **–** Donnez les avantages et les inconvénients des implémentations pour les deux fonctions *tableauListe* et *creuxDeListe.*

**Question 4.** Les internautes utilisent un logiciel pour naviguer sur Internet. Parmi ses fonctions, il y a deux fonctions symbolisées « ⇐ » « ⇒ » qui permettent de naviguer au sein des pages (URL). Par exemple, si l'utilisateur consulte dans l'ordre les pages d'adresses U1,U2,U3,U4, lors de l'affichage de U4, « ⇐ » permet de revenir à U3 et on peut ensuite revenir à U4 par « ⇒ ». Si depuis U3 on visualise une page U5 alors l'accès à U4 est perdu. On rappelle que l'adresse d'une page est une chaine de caractère.

1 - Quelle structure de données peut-on utiliser dans le navigateur pour mémoriser les adresses des pages ? Pourquoi ?

2 - Quelle structure de données peut-on utiliser dans le navigateur pour réaliser les fonctions« ⇐ » et « ⇒ »? Pourquoi ?

3 - Décrivez le(s) type(s) abstrait(s) correspondant à ces structures et notamment le champ valeur.

4 - Ecrire les primitives d'accès et de modification de ce(s) type(s).

5 – Une autre fonctionnalité du navigateur consiste à proposer dans la zone de saisie de l'URL, une liste d'URL au fur et à mesure que l'utilisateur écrit des caractères (complétion). Quelle structure de donnée peut permettre de gérer cette fonction ? Pourquoi ? Modifier le type abstrait de la question 3 en conséquence. On ne demande pas d'écrire les primitives.

**Listes simplement chainées (listeSC)**

fonction valeur(val L:liste d'objet):objet;
fonction debutListe(val L:liste d'objet) :vide ;
fonction suivant(val L:liste d'objet) :vide ;
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;

**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;

**File de priorité**

fonction changeValeur(ref T:tas d'objet,val s:sommet,val v:objet):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;

*FIN*