Université Bordeaux 1 Licence Informatique 2011-2012

Algorithmique 1
Contrôle continu 4

Exercice 1 *Question de cours*
Dire si les affirmations suivantes sont vraies ou fausses, en justifiant (brièvement) votre réponse.

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

Exercice 2 *Tas*

1. On considère la suite de clés $S = (10, 2, 14, 12, 7, 8, 5, 18)$.
   - Construire le tas max correspondant à l'insertion successive des clés de S. On dessinera le tas après insertion de $(10, 2, 12)$ puis après insertion de chacune des clés suivantes.
   - Montrer l'exécution de la fonction `supprimer` sur le tas max obtenu.
2. Ecrire une fonction `tasToListe` qui construit une liste simplement chaînée `L` contenant tous les éléments d'un tas max dans l'ordre croissant. Quelle est sa complexité ?

Exercice 3 *Arbres binaires de recherche*

1. Si S est un sommet possédant un fils droit d'un arbre binaire de recherche, où se trouve le sommet *successeur* du sommet S (sommet dont la valeur est la suivante de la valeur de S dans l'ordre croissant) ?
2. Ecrire une fonction `successeur` qui retourne le sommet successeur de S.
3. Ecrire une fonction `supprimeSuccesseur` qui supprime le successeur de S et qui renvoie sa valeur. (on suppose que S a un successeur)

Annexe A **Type abstrait** *arbreBinaire*

```
arbreBinaire= curseur;
sommet= curseur;
```

- **Création**
  `fonction creerArbreBinaire(val Racine:objet):sommet;`
- **Accès**
  `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 estFeuille(val S:sommet):booleen;`
- **Modification**
  `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;`

Annexe B **Type abstrait** *tas.*

```
fonction valeur(val T: tas d'objet): objet;
fonction ajouter(ref T: tas d'objet, val v: objet): vide;
fonction supprimer(ref T: tas d'objet): vide;
fonction creerTas(ref T: tas d'objet, val v: objet): vide;
fonction detruireTas(ref T: tas d'objet): vide;
```