Contrôle continu 4
(3 pages)

Questions de cours (5 points)

- Dans une table de hashage, pourquoi peut-on ne pas avoir accès à une clé en O(1)
  - dans le cas d'adressage ouvert ?
  - dans le cas d'adressage chainé ?
- Peut-on avoir un arbre binaire de recherche dont les sommets sont des tas min? Si oui écrire une fonction permettant de comparer deux tas. Si non, expliquez pourquoi.
- On considère une table de hashage munie la fonction de hashage sur la clé x

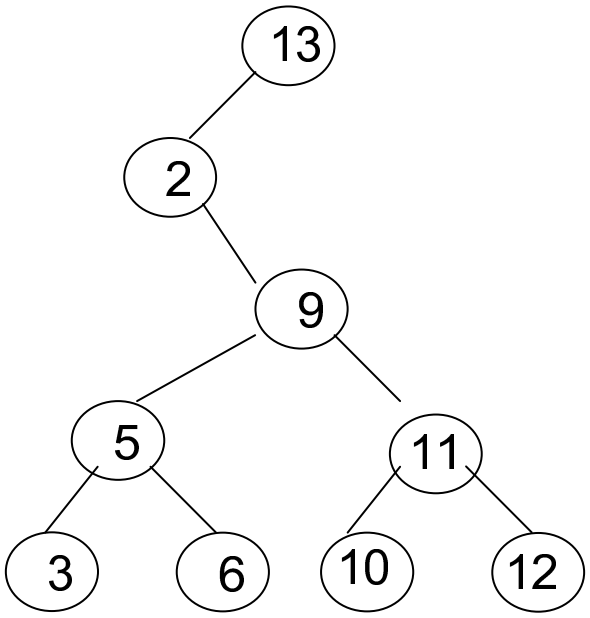
$$h(x) = (3x+1) \bmod M$$

  - Quelle sont les index possibles pour le tableau de la table de hashage ?
  - Quel problème génère le choix de M=12 ?
  - On suppose que M=13, dessinez la structure de données, après la suite d'insertion suivante **(**23,13,8,12,27,9,29,36,3**)** dans le cas de l'adressage chainé.
  - On suppose que M=13, dessinez la structure de données, après la suite d'insertion suivante **(**23,13,8,12,27,9,29,36,3**)**. On utilisera la fonction de sondage

$$h(x) = (5x+1) \bmod M$$

Exercice 2 (5 points)

- Soit l'arbre binaire de recherche suivant



Pour chaque question, on considère que l'arbre à modifier est celui de la figure ci-dessus. On donnera les détails des opérations.

1. Donnez l'arbre après insertion de 7
2. Donnez l'arbre après suppression de 2
3. On considère la suite des clés dans l'ordre hiérarchique construire le tas max correspondant.
4. Equilibrez l'arbre binaire de recherche, on donnera les étapes et les rotations.

5. On considère la suite des clés dans l'ordre préfixe construire le 2-3 arbre correspondant.

Exercice 2 (10 points)

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 identificateurs des 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 en détaillant leurs éléments.

4 – Ecrire les fonctions *ajouterCommande* (utilisée par R1) et *supprimerCommande* utilisé par R2.

5 – Modifier la fonction *ajouterCommande* pour éviter les doublons.

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;