|  |  |  |
| --- | --- | --- |
| Description : Afficher l'image d'origine | **Université Cadi Ayyad**  **Faculté des sciences Semlalaia Marrakech**  **Département d'informatique** |  |

**Structure de données Examen 2017/2018 (Durée 2h)**

**Exercice 1 : file d’attente au cinéma (3,5 points)**

Considérons une file d’attente devant un cinéma. La file initialement vide se remplit au fur et à mesure que les individus arrivent avec une gestion particulière liée au fait que si un nouvel individu aperçoit dans la file un ami, alors il se joint à lui pour attendre.

Pour manipuler cette liste d’attente, vous considérerez que les individus sont représentés par des entiers. Deux amis seront alors deux entiers identiques. La structure de données utilisée pour représenter la liste devra donc intégrer non seulement l’individu, mais aussi le nombre d’occurrences associé.

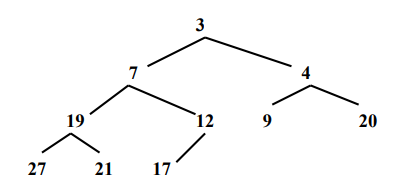
1. Proposez une structure de données permettant de gérer une telle file (0,5 pt)
2. Ecrire la fonction permettant d’ajouter un nouvel individu dans une telle file (2 pts)
3. Ecrire la fonction permettant de retirer le premier individu d’une telle file (1 pts)

**Exercice 2 : Arbre binaire de recherche et AVL (8,5 points)**

1. Définir une structure **struct noeud\_s** permettant de coder un nœud d'un arbre binaire contenant une valeur entière. Ajouter des **typedef** pour définir les nouveaux types **noeud\_**t et **arbre\_t**. (0,5 pt)
2. Écrire une fonction **cree\_arbre()** qui prend en argument une valeur entière ainsi que deux arbres et renvoie un arbre dont la racine contient cette valeur et les deux sous-arbres sont ceux donnés en paramètre. (1 pt)
3. Écrire une fonction (récursive) **detruit\_arbre()** qui libère la mémoire occupée par tous les nœuds d'un arbre binaire. (1 pt)
4. Écrire une fonction (récursive) **nombre\_de\_noeuds()** qui calcule le nombre de nœuds d'un arbre binaire. (1 pt)
5. Écrire une fonction **affiche\_arbre()** qui affiche les valeurs des nœuds d'un ABR par ordre croissant (choisissez le bon type de parcours des nœuds de l'arbre. . . ). (1 pt)
6. Écrire une fonction (récursive) **insère ()** qui ajoute une valeur dans l'ABR (ce sera un nouveau nœud placé correctement dans l'arbre). (1 pts)
7. Dessinez l’arbre binaire de recherche obtenu par ajout successif aux feuilles des éléments suivants : **10 ; 12 ; 7 ; 14 ; 16 ; 15 ; 4 ; 9 ; 8 ; 5 ; 3 ; 2**
   1. Supprimer les éléments **15 ; 9 et 7** en utilisant l’algorithme vu en cours. (1 pt)
   2. L’arbre obtenu est-il un **AVL** ? Si non, transformez-le en AVL en effectuant des rotations. Justifier les rotations effectuées (2 pts).

**Exercice 3 TAS (2 points)**

Soit l’arbre suivant :



1. Cet arbre s’agit-il d’un **TAS min** ? justifier votre réponse ? (0,5 pt)
2. Insérer la valeur 5. Décrire les étapes d’insertion ? (0,75 pt)
3. Supprimer la valeur 3. Décrire les étapes de suppression ? (0,75 pt)

**Exercice 4 (2 pts)**

**Table de hachage: gestion des collisions par adressage ouvert**

1. On considère la fonction de hachage: **h(c,i) = (h'(c)+ i) mod m** avec **h'(c)= c mod m** avec **m =11**.Insérez successivement les clés **11, 22, 31, 4, 15, 28, 17, 88 et 59** dans la table en utilisant un sondage linéaire. (1 pt)

**Table de hachage: gestion des collisions par chaînage.**

1. Même question en résolvant les collisions par chaînage. (1 pt)

**Exercice 5 Le tri rapide (Quick Sort) (2 points)**

Soit le tableau tab suivant des entiers dans un ordre quelconque :



Appliquer l'algorithme de **tri-rapide** pour trier les entiers par ordre croissant.

Au départ, tri-rapide (**tab, 1,11**)

**Pivot** =de votre choix (**valeur médiane, première valeur , etc….**)

**Indices** : i=**1**, j=**11**

**Exercice 6 Graphe (2 points)**

Indiquer l’ordre de parcours des sommets du graphe ci-dessous dans un parcours en profondeur **dfs** à partir du sommet **1**? Vous donnerez à chaque fois l'ordre d'entrée des sommets dans la **pile**.

