Université Ibn Tofail Faculté des sciences Département d'Informatique	**Examen : Programmation II** **SMI/S4** **2017/2018**
Session : Printemps (Normale)	**Durée : 1h30**

Exercice 1 : (10 points)

On souhaite écrire un programme permettant de gérer l'ensemble des livres d'une bibliothèque. Pour chaque livre, on conserve les informations suivantes :

- le code du livre (**code : long**),
- le titre du livre (**titre : chaine de caractères**),
- le nom de l'auteur (**auteur : chaine de caractères**),
- le nombre d'exemplaires (**nbExpl : long**).

1. Définir le type **Livre** qui représente correctement un livre. **(1 pts)**
2. Définir une fonction **void initLivre(…)** qui permet d'initialiser le code, le titre, l'auteur et le nombre d'exemplaire d'un livre. **(1 pts)**
3. Définir une fonction **void afficherLivre(…)** qui permet d'afficher les données d'un livre sous forme : **code titre nom de l'auteur nombre exemplaire (1 pts)**

On souhaite également gérer l'ensemble des livres de la bibliothèque contenant un nombre quelconque de livres (*non prédéfini*) en créant une structure permettant de faciliter le classement des livres dans la bibliothèque.

4. Définir la structure **Noeud** d'une liste chainée représentant la bibliothèque, chaque **Noeud** représente un livre de la bibliothèque avec un suivant (*adresse*) vers un autre **Noeud**. **(1 pts)**
5. Définir une fonction **int existeLivre(Noeud * bib, Livre liv)** qui permet de vérifier si un livre existe dans une bibliothèque, ou non. La fonction retourne **1** si le livre existe dans la liste, **0** sinon. **(1,5 pts)**
6. Définir une fonction **void ajouterLivre(Noeud * bib, Livre liv)** qui permet d'ajouter un livre à la fin de la liste si le livre n'existe pas dans la liste, sinon il incrémente le nombre d'exemplaires de ce livre. **(1,5 pts)**
7. Définir une fonction **int nbrLivresAuteur(Noeud * bib, char auteur[])** qui permet de renvoyer le nombre de livres d'un auteur donné. **(1,5 pts)**
8. Définir une fonction **Noeud * fusionnerBibliothèque(Noeud * bib1, Noeud * bib2)** qui permet de renvoyer la fusion de deux listes de livre (bibliothèques) passés en paramètre. La fusion résultante ne doit pas comporter de doublant (c.-à-d. livre dupliqué). **(1,5 pts)**

Exercice 2 : (6 points)

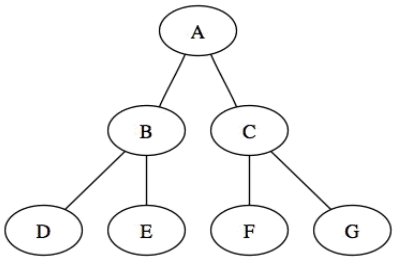
Soit un fichier de données texte "**etudiants.txt**" structuré en une suite de lignes contenant chacune le nom de l'étudiant, le prénom de l'étudiant, le semestre d'inscription de l'étudiant et la moyenne obtenue. La première ligne contenant le nombre d'étudiants enregistré dans le fichier.

4			
Hamid	**Tahiri**	**S1**	**15**
Merieme	**Karimi**	**S1**	**10**
Aicha	**El Gharbaoui**	**S3**	**7**
Ali	**Mhamdi**	**S2**	**11**

1. Créer la fonction **void listerEtudiants(char fichEtudiants[])** : qui permet de récupérer l'ensemble des étudiants enregistrés dans le fichier "**etudiants.txt**"et les afficher. **(1,5 pts)**
2. Créer une structure **Etudiant** représentant les différentes informations figurant dans chaque ligne du fichier texte. **(1 pts)**
3. Créer la fonction **void enregisterEtudiant(Etudiant e, char fichEtudiant[])**qui permet d'enregistrer l'étudiant passé en paramètre dans le fichier **"etudiants.txt". (1,5 pts)**
4. Donner la fonction **void lireEtudiants(char fichEtudiant[], Etudiant * tabEtu)** qui permet de récupérer l'ensemble des étudiants enregistrés dans le fichier **"etudiants.txt"** et les stocker dans un tableau de structure **tabEtu**. **(2 pts)**

Exercice 3 : (4 points)

Soit l'arbre suivant :



1. Donner la structure **Nœud** représentant les éléments de cet arbre. **(1 pts)**
2. Quel est le type de cet arbre ? et exprimer le nombre de nœuds **n** de ce type d'arbre en fonction de la hauteur **h. (1,5 pts)**
3. Créer une fonction **parcoursLargeur()** qui permet de parcourir cet arbre en largeur et afficher ses éléments. Le résultat doit être l'affichage suivant : A B C D E F G. **(1,5 pts)**