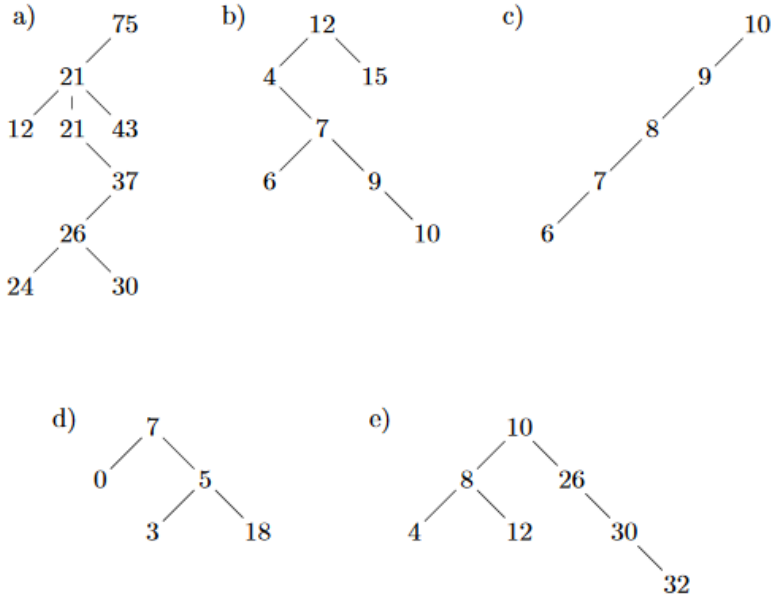
Université Ibn Tofail Faculté des sciences Département d'Informatique	**Examen : Programmation II** **SMI/S4** **2017/2018**
Session : Printemps (Rattrapage)	**Durée : 1h30**

Exercice 1 : (8 points)

1. Donner la structure **Nœud** représentant les éléments d'un arbre binaire contenant une valeur entière. (1pts)
2. Donner une fonction qui prend un arbre binaire en entrée et renvoie son nombre de nœuds ayant un seul fils. (1.5pts)
3. Lesquels de ces arbres ne sont pas des arbres binaires de recherche? Justifier brièvement. (1.5pts)



4. Soit A un pointeur sur un arbre binaire de recherche vide, Dessiner l'arbre obtenu en insérant successivement 1, 4, 2, 11, 19, 5, 3, 10 et 8. (1pts)
5. Donner une fonction qui prend un arbre binaire en entrée et qui renvoie 1 si c'est un arbre binaire de recherche et 0 sinon. (1.5pts)
6. Ecrire une fonction qui affiche les valeurs des nœuds d'un arbre binaire de recherche en ordre croissant (choisissez le bon type de parcours des nœuds de l'arbre…) (1.5pts)

Exercice 2 : (12 points)

Nous allons considérer deux structures imbriquées :
* une structure de données DATE, qui contiendra comme champs 3 entiers (jour, mois, année),
* une structure de données PERSONNE ayant pour données :
   - un pointeur sur une chaîne de caractères « nom », qui correspondra au nom de la personne,
   - un tableau « telephone » de 10 caractères, qui correspondra au numéro de téléphone de la personne,
   - une structure de données de type DATE, qui correspondra à la date d'inscription dans la liste chaînée.
   - un pointeur sur struct personne, nommé « suivant », qui pointera sur l'élément suivant de la liste chaînée.

1. Écrire les déclarations des structures PERSONNE et DATE. **(2pts)**

2. Écrire une fonction **PERSONNE* AllouePersonne()** qui alloue la mémoire nécessaire pour un élément de type **PERSONNE**. **(1pts)**
3. Écrire une procédure **void SaisiePersonne(PERSONNE *pers)** qui permet à l'utilisateur de saisir au clavier les différents champs d'un élément de type **PERSONNE** que l'on fera passer par adresse comme argument d'entrée. **(1pts)**
4. Écrire une fonction **RecherchePersonne** pour chercher une personne dans une liste chaînée selon son nom (la liste et le nom à chercher seront passés en argument d'entrée) qui renverra l'adresse de la structure correspondant au nom ou NULL si le nom n'appartient pas à la liste chaînée. **(1pts)**
5. Écrire une fonction **PERSONNE *InserePersonne(PERSONNE *tete, PERSONNE *pers)** qui permet d'insérer une nouvelle personne (représentée par une structure de type **PERSONNE** que l'on fera passer à la fonction par adresse) dans la liste chainée ; cette insertion devra se faire de sorte à :
   - ce que la liste chaînée soit triée selon l'ordre alphabétique du champ « nom »
   - ce qu'aucun élément ne soit en double (même nom, même numéro de téléphone) dans la liste chaînée résultante. **(1.5pts)**
6. Écrire une fonction qui enregistre les personnes de la liste chainée dans un fichier texte de manière à avoir une personne par ligne. La tête de la liste et le nom du fichier sont passés en arguments d'entrée à la fonction. **(1.5pts)**

Nous désirons gérer un annuaire de personne en définissant un tableau d'adresse où chaque case du tableau pointe sur la tête d'une liste chaînée ne contenant que des personnes dont le nom commence par la même lettre (en majuscules ou en minuscules). Pour gérer ce tableau comme une pile, nous allons considérer une structure de données **struct annuaire** (de type synonyme **ANNUAIRE**) contenant :

- un double pointeur **tab** sur des **struct personne** qui pointera sur le tableau d'adresses des têtes des listes chaînées de type **struct personne**.
- un entier **Max** qui correspondra au nombre maximum de listes chaînées que peut contenir le tableau.
- un entier **Nb** qui représentera le nombre de listes chaînées contenues.

```
typedef struct annuaire
{
  PERSONNE **tab ;
  int Max ;
  int Nb ;
} ANNUAIRE ;
```

7. Écrire une fonction **AllouеAnnuaire** qui alloue et renvoie l'adresse sur une structure de type struct annuaire ; cette fonction devra aussi allouer le champ tab de sorte à contenir au maximum 26 cases (il n'y a que 26 lettres dans l'alphabet) ainsi que donner des valeurs initiales aux autres champs. **(1pts)**
8. Écrire une fonction **InserePersonnesAnnuaire** qui permet d'insérer une liste chaînée de type **struct personne** dans une structure de type **ANNUAIRE** (les deux ayant été passés par adresse en argument d'entrée). Cette fonction devra fractionner la liste chaînée si elle contient des personnes aillant des noms commençant par des lettres différentes. Nous rappelons que l'annuaire doit être géré comme une pile**. (1.5pts)**
9. Écrire une fonction **RecherchePersonneAnnuaire** qui permet de rechercher une personne dans l'annuaire. A partir du nom passé en argument d'entrée, elle devra renvoyer un pointeur sur la fiche correspondant à la personne ou la valeur NULL si ce nom n'appartient pas à l'annuaire. **(1.5pts)**