Institut Préparatoire aux Etudes d'Ingénieurs de Monastir

Première année préparatoire

Mathématiques et Physique, Physique et Chimie et Technologie

Informatique

Examen 1

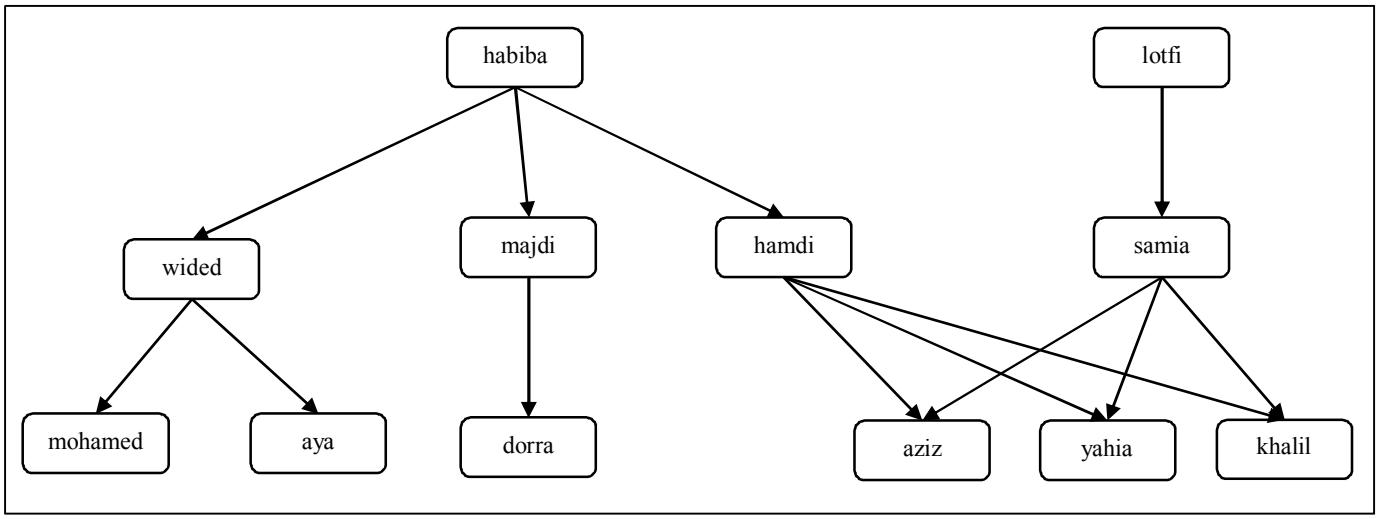
Date : 12/12/2022	**Heure : 10:30h**	**Durée : 2 H**	**Nombre de pages : 3**
Barème : Exercice : 7 points	**Problème : 13 points**		

Exercice:

Nous supposons qu'une famille est représentée par un dictionnaire dont les éléments sont des associations entre le prénom de la personne et l'ensemble des prénoms de ses enfants. On suppose que :

- toutes les personnes de cette famille possèdent des prénoms différents.
- si une personne n'a pas d'enfants, alors elle ne figure pas comme clé dans le dictionnaire.

Exemple **:** Soit la famille **F** représentée par l'arborescence suivante :



Cette famille est définie par **F** = {'habiba': {'majdi', 'wided', 'hamdi'}, 'hamdi': {'khalil', 'yahia', 'aziz'}, 'lotfi': {'samia'}, 'majdi': {'dorra'}, 'samia': {'aziz', 'khalil', 'yahia'}, 'wided': {'mohamed', 'aya'}}

Dans cet exemple :

- les enfants de samia sont aziz, khalil et yahia.
- les petits-enfants de lotfi sont aziz, khalil, yahia.
- mohamed n'a pas d'enfants.
- Les parents de khalil sont hamdi et samia
- Les frères et sœurs de majdi sont wided et hamdi.
- Les neveux et nièces de hamdi sont dorra, mohamed et aya.
- Les oncles et tantes de aziz sont wided et majdi.

Ecrire en PYTHON les fonctions suivantes :

1. la fonction **enfant** qui prend en paramètre une famille **F** et un prénom **p** et retourne l'ensemble des prénoms des enfants de **p** dans **F**. On retourne un ensemble vide si **p** n'a pas d'enfants.
2. la fonction **ens_enfant** qui prend en paramètre une famille **F** et un ensemble **X** contenant des prénoms et retourne l'ensemble des prénoms de tous les enfants des personnes de l'ensemble **X**.
3. la fonction **petit_enfant** qui prend en paramètre une famille **F** et un prénom **p** et retourne l'ensemble des prénoms des petits-enfants de **p**.
4. la fonction **parent** qui prend en paramètre une famille **F** et un prénom **p** et retourne l'ensemble des prénoms des parents de **p**.
5. la fonction **freres_soeurs** qui prend en paramètre une famille **F** et un prénom **p** et retourne l'ensemble des prénoms des frères et sœurs de **p**.
6. la fonction **neveux_nieces** qui prend en paramètre une famille **F** et un prénom **p** et retourne l'ensemble des prénoms des neveux et des nièces de **p** dans **F**. (neveu et nièce sont les enfants des frères et sœurs).
7. la fonction **oncles_tantes** qui prend en paramètre une famille **F** et un prénom **p** et retourne l'ensemble des prénoms des oncles et des tantes de **p**.

Problème :

Ce problème est composé de trois parties dépendantes et doit être traité en PYTHON.

PREMIERE PARTIE

On définit un logiciel par un dictionnaire contenant les clés (de type str) suivantes :

Clé	Type de valeur correspondante
'Designation'	chaine formée au maximum de 20 caractères
'Licence'	caractère ('**L**' **ou** '**l**' : Libre– '**C**' **ou** '**c**' : Commercial –'**P**' **ou** '**p**' : Partagiciel)
'Editeur'	chaine (ce champ représente le nom de l'éditeur du logiciel)
'AnneeEdition'	entier (ce champ représente l'année de l'édition du logiciel)
'Version'	chaine (ce champ représente la version du logiciel)
'TailleDisque'	entier (ce champ représente la taille nécessaire sur le disque en Kilo octets.)

On se propose de gérer une liste de logiciels enregistrée sur un ordinateur.

En se basant sur les représentations proposées, écrire en Python les fonctions suivantes :

1. **SaisieLogiciel** : une fonction permettant de retourner un logiciel saisi à partir du clavier.
2. **AjoutLogiciel** : une fonction qui prend en paramètre une liste de logiciels **L** et un logiciel **X** et permet d'ajouter **X** à la fin de **L**.
3. **ListeLogicielsParEditeur :** une fonction qui prend en paramètre une liste de logiciels **L** et le nom de l'éditeur **e** et retourne un tuple de logiciels développés par l'éditeur **e.**
4. **EspaceTotalOccupe :** une fonction permettant de calculer l'espace disque total occupé par tous les logiciels de la liste de logiciels **L** passée en paramètre.

5. **RechercheLogiciel** : une fonction qui prend en paramètre une liste de logiciels **L** et permet de retourner le rang d'un logiciel dans la liste **L** connaissant sa désignation **d** donnée en paramètre. Si le logiciel n'existe pas, on retourne -1.
6. **NombreLogicielCommercial** : une fonction qui prend en paramètre une liste de logiciels **L** et retourne le nombre de logiciels existants dans la liste **L** qui ont une licence commerciale.

DEUXIÈME PARTIE

Maintenant, on définit un ordinateur par un dictionnaire contenant les clés (de type str) suivantes :

Clé	Type de valeur correspondante
'Num'	entier (ce champ représente le numéro de l'ordinateur)
'Marque'	chaine (ce champ représente la marque de l'ordinateur)
'CapaciteDisque'	entier (ce champ représente la taille de disque dur en Kilo octets)
'LogInst'	list (ce champ représente la liste des logiciels installés sur cet ordinateur)

Implémenter, en python, les fonctions suivantes :

7. **EspaceOccupe** : qui prend en paramètre un ordinateur **O** et retourne en Kilo octets l'espace occupé par tous les logiciels installés sur l'ordinateur **O**.
8. **InstallerLogiciel** : qui prend en paramètre un ordinateur **O** et un logiciel **X** et permet d'ajouter **X** à la liste de logiciels installés sur l'ordinateur **O**. On ne peut ajouter le logiciel que lorsque l'espace libre de l'ordinateur est supérieur ou égal à la taille du logiciel sur le disque.
   N.B : espace libre = capacité Disque – espace occupé par les logiciels installés.
9. **DesinstallerLogiciel** : qui prend en paramètre un ordinateur **O** et une désignation **d** d'un logiciel et permet de supprimer un logiciel, dont la désignation est **d**, de la liste de logiciels qui sont installés sur l'ordinateur **O**.
10. **ViderOrdinateur** : permet de désinstaller tous les logiciels d'un ordinateur **O** passé en paramètre.

TROISIÈME PARTIE

Dans cette partie, nous nous intéressons à la gestion d'un parc informatique formé par un ensemble d'ordinateurs. **Chaque ordinateur est identifié par son numéro dans le parc (identifiant unique)**.
Cet ensemble est modélisé par une liste.

Implémenter, en Python, les fonctions suivantes :

11. **AjoutOrdinateur** : qui prend en paramètre un ordinateur **O** et une liste d'ordinateurs **LO** et permet d'ajouter **O** au début de la liste **LO**.
12. **SupprimerOrdinateur** : qui prend en paramètre une liste d'ordinateurs **LO** et un numéro **N** d'un ordinateur et permet de supprimer l'ordinateur de numéro **N** de la liste d'ordinateurs **LO**.
13. **FormaterOrdinateur :** qui prend en paramètre une liste d'ordinateurs **LO** et un numéro **N** d'un ordinateur, et permet de supprimer tous les logiciels de l'ordinateur de numéro **N**.