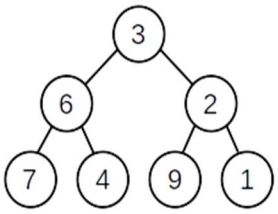
EXERCICE 4 (4 points)

Cet exercice, composé de deux parties A et B, porte sur le parcours des arbres binaires, le principe "diviser pour régner" et la récursivité.

Cet exercice traite du calcul de la somme d'un arbre binaire. Cette somme consiste à additionner toutes les valeurs numériques contenues dans les nœuds de l'arbre.

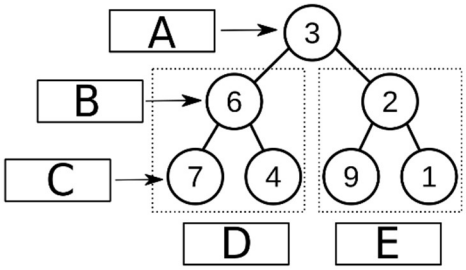
L'arbre utilisé *dans les parties A et B est le suivant :*



Partie A : Parcours d'un arbre

1. Donner la somme de l'arbre précédent. Justifier la réponse en explicitant le calcul qui a permis de l'obtenir.

2. Indiquer la lettre correspondante aux noms 'racine', 'feuille', 'nœud', 'SAG' (Sous Arbre Gauche) et 'SAD' (Sous Arbre Droit). Chaque lettre **A**, **B**, **C**, **D** et **E** devra être utilisée une seule fois.



Arbre avec les lettres à associer

3. Parmi les quatre propositions A, B, C et D ci-dessous, donnant un parcours en largeur d'abord de l'arbre, une seule est correcte. Indiquer laquelle.
Proposition A : 7 - 6 - 4 - 3 - 9 - 2 - 1
Proposition B : 3 - 6 - 7 - 4 - 2 - 9 - 1
Proposition C : 3 - 6 - 2 - 7 - 4 - 9 - 1
Proposition D : 7 - 4 - 6 - 9 - 1 - 2 – 3

4. Écrire en langage Python la fonction `somme` qui prend en paramètre une liste de nombres et qui renvoie la somme de ses éléments.
Exemple : `somme([1, 2, 3, 4])` est égale à 10.

5. La fonction `parcourir(arbre)` pourrait se traduire en langage naturel par :

```
parcourir(A):
    L = liste_vide
    F = file_vide
    enfiler A dans F
    Tant que F n'est pas vide
        défiler S de F
        ajouter la valeur de la racine de S dans L
        Pour chaque sous arbre SA non vide de S
            enfiler SA dans F
    renvoyer L
```

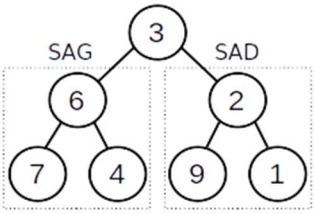
Donner le type de parcours obtenu grâce à la fonction `parcourir`.

Partie B : Méthode 'diviser pour régner'

6. Parmi les quatre propositions A,B, C et D ci-dessous, indiquer la seule proposition correcte.

En informatique, le principe diviser pour régner signifie :
Proposition A : diviser une fonction en deux fonctions plus petites
Proposition B : utiliser plusieurs modules
Proposition C : séparer les informations en fonction de leur types
Proposition D : diviser un problème en deux problèmes plus petits et indépendants.

7. L'arbre présenté dans le problème peut être décomposé en racine et sous arbres :



Indiquer dans l'esprit de 'diviser pour régner' l'égalité donnant la somme d'un arbre en fonction de la somme des sous arbres et de la valeur numérique de la racine.

8. Écrire en langage Python une fonction récursive `calcul_somme(arbre)`. Cette fonction calcule la somme de l'arbre passé en paramètre.

Les fonctions suivantes sont disponibles :

- `est_vide(arbre)` : renvoie `True` si `arbre` est vide et renvoie `False` sinon ;
- `valeur_racine(arbre)` : renvoie la valeur numérique de la racine de `arbre` ;
- `arbre_gauche(arbre)` : renvoie le sous arbre gauche de `arbre` ;
- `arbre_droit(arbre)` : renvoie le sous arbre droit de `arbre`.