EXERCICE 2

Cet exercice porte sur de l'algorithmique et de la programmation en langage Python utilisant les structures de données du type arbre binaire.

On crée un jeu dans lequel les personnages doivent se déplacer. Le personnage Mélusine se déplace et se retrouve devant différents chemins possibles. Le schéma 1 ci-dessous illustre la situation.

Mélusine part du point A et peut se rendre à n'importe quel autre point repéré par une lettre. Sous chaque lettre, se cache un objet à récupérer. Chaque objet est associé à une valeur.

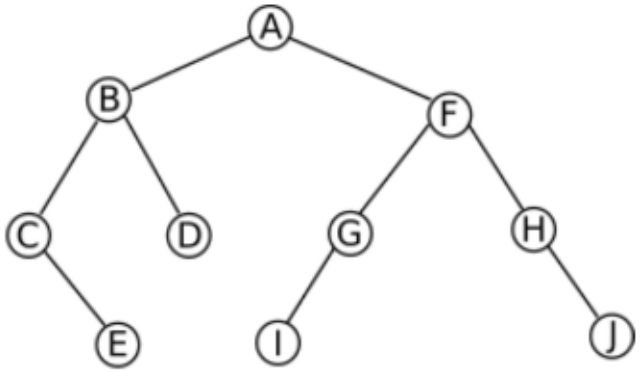

Schéma 1

1) **Indiquer** pourquoi il s'agit d'un arbre binaire.

2) À chaque lettre, on attribue une valeur positive. On définit la variable `V` suivante contenant les valeurs associées aux lettres :

```
V = {'A': 1 , 'B': 2 , 'C': 3 , 'D': 5 , 'E': 10 , 'F': 15 , 'G': 4 , 'H': 5 , 'I': 5 , 'J': 7}
```

   a) **Indiquer** le type de la variable V.

   b) **Écrire** l'instruction permettant d'accéder à la valeur « 7 » stockée dans cette variable.

   c) **Écrire** une fonction `somme` en langage Python qui prend un paramètre `W` du même type que la variable `V` et qui renvoie la somme des valeurs de `W`.

   d) **Écrire** une fonction `VMax` en langage Python qui prend un paramètre `W` du même type que la variable `V` et qui renvoie la lettre ayant la valeur maximale de `W`.

3) **Indiquer**, en **justifiant** la réponse, le rôle de l'algorithme suivant.

```
CALCUL(T : arbre) :
VARIABLE
x : noeud
DEBUT
  si T n'est pas un arbre vide :
    x ← racine de T
    renvoyer 1 + CALCUL(sous arbre gauche de x) + CALCUL(sous arbre
droit de x)
  sinon :
    renvoyer 0
  fin si
FIN
```

4) On applique l'algorithme ci-dessous à l'arbre du schéma 1.

```
VISITE(T: arbre) :
VARIABLE
x : noeud
DEBUT
  si T n'est pas un arbre vide :
    x ← racine de T
    affiche clé de x
    VISITE(sous arbre gauche de x)
    VISITE(sous arbre droit de x)
  fin si
FIN
```

a) **Indiquer** l'affichage obtenu.

b) **Indiquer** le type de parcours réalisé par l'algorithme `VISITE`.