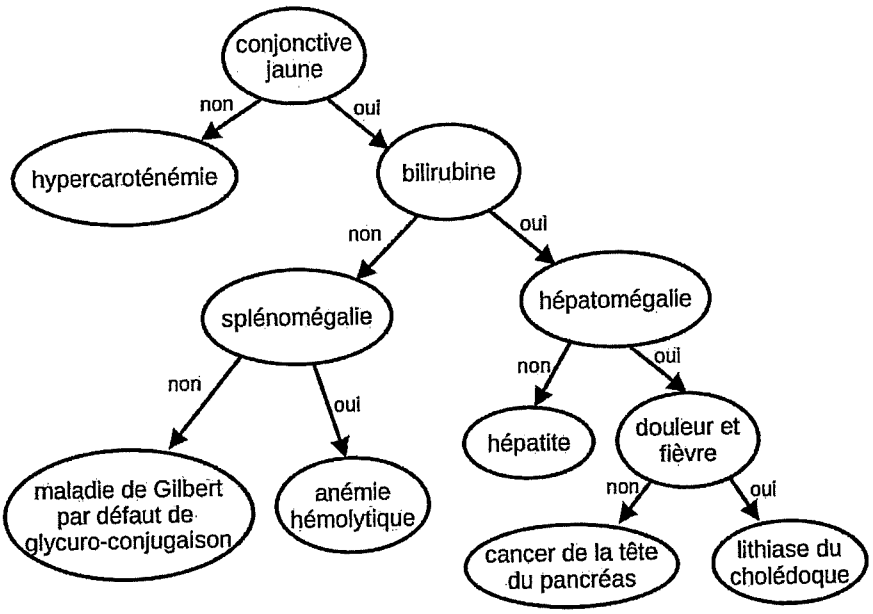
Exercice 3 (4 points)

Cet exercice porte sur les arbres binaires.

Les premiers travaux concernant l'aide à la décision médicale se sont développés pendant les années soixante-dix parallèlement à l'avènement de l'informatique dans le secteur médical. L'arbre de décision est une technique décisionnelle fréquemment employée pour rechercher la meilleure stratégie thérapeutique. L'arbre de décision de cet exercice, présenté ci-dessous, est un arbre binaire que l'on nommera `arb_decision`.



Arbre de décision en présence d'une jaunisse (peau anormalement jaune) chez un patient.

Rappels :

- ✓ Un **arbre binaire** est une structure de données qui peut se représenter sous la forme d'une hiérarchie dont chaque élément, appelé **nœud**, porte une étiquette.
- ✓ Le nœud initial est appelé **racine**.
- ✓ Chaque nœud d'un arbre binaire possède au plus deux **sous-arbres**.
- ✓ Chacun de ces sous-arbres est un arbre binaire, appelés sous-arbre gauche et sous-arbre droit.
- ✓ Un nœud dont les sous-arbres sont vides est appelé une **feuille**.
- ✓ Dans cet exercice, on utilisera la convention suivante : la **hauteur** d'un arbre binaire ne comportant qu'un nœud est égale à **1**.

Dans l'arbre de décision en présence d'une jaunisse chez un patient,

- ✓ un **nœud** représente un symptôme dont le médecin doit étudier la présence ou l'absence ; la réponse ne peut être que **oui** ou **non** ;
- ✓ le sous-arbre gauche d'un nœud donné décrit la démarche à adopter si le symptôme est **absent** ;
- ✓ le sous-arbre droit d'un nœud donné décrit la démarche à adopter si le symptôme est **présent** ;
- ✓ l'étiquette d'une feuille est la maladie induite par le chemin parcouru.

1. Déterminer la taille et la hauteur de l'arbre donné en exemple en introduction (arbre de décision en présence d'une jaunisse).

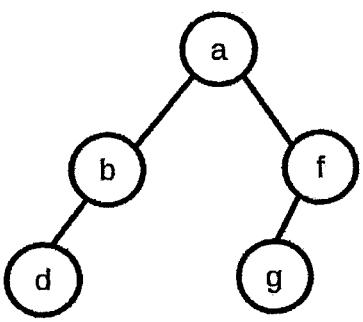
2. On choisit d'implémenter un arbre binaire à l'aide d'un dictionnaire.

```
arbre_vide = {}
arbre = {'etiquette': 'valeur' ,
         'sag': sous_arbre_gauche ,
         'sad': sous_arbre_droit }
```

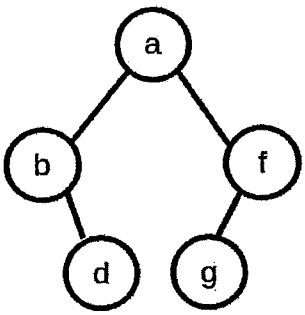
Le code ci-dessous représente un arbre selon le modèle précédent.

```
{'etiquette' :'a',
 'sag':{'etiquette' :'b',
        'sag':{},
        'sad' : {'etiquette' : 'd',
                 'sag' : {},
                 'sad' : {}}},
 'sad': {'etiquette' :'f',
         'sag' : {'etiquette' : 'g',
                  'sag' : {},
                  'sad' : {}},
         'sad' : {} }}
```

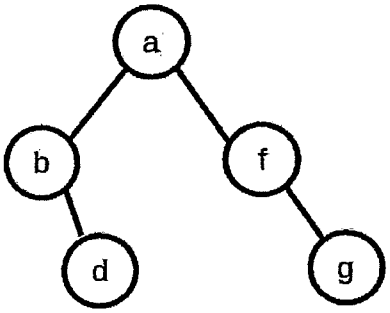
a. À quelle représentation graphique correspond la structure implémentée ci-dessus ?



arbre 1



arbre 2



arbre 3

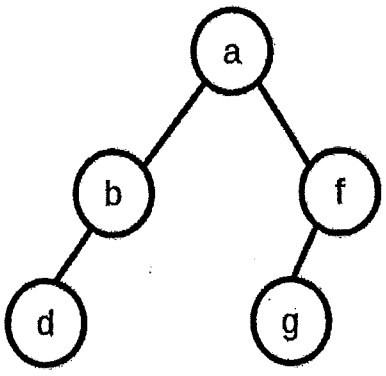
b. Représenter graphiquement l'arbre correspondant au code ci-dessous.

```
{'etiquette' :'H',
 'sag':{ 'etiquette' :'G',
         'sag': {'etiquette' : 'E',
                 'sag' : {},
                 'sad' : {}},
         'sad' : {'etiquette' : 'D',
                  'sag' : {},
                  'sad' : {'etiquette' : 'B',
                           'sag' : {},
                           'sad' : {} }}},
 'sad': {'etiquette' :'F',
         'sag' : {'etiquette' : 'C',
                  'sag' : {},
                  'sad' : {'etiquette' : 'A',
                           'sag' : {} ,
                           'sad' : {}}},
          'sad' : {} }}
```

3. La fonction `parcours(arb)` ci-dessous permet de réaliser le parcours des nœuds d'un arbre binaire `arb` donné en argument.

```
def parcours(arb):
    if arb == {}:
        return None
    parcours(arb['sag'])
    parcours(arb['sad'])
    print(arb['etiquette'])
```

a. Donner l'affichage après l'appel de la fonction `parcours` avec l'arbre dont une représentation graphique est ci-contre.



b. Écrire une fonction `parcours_maladies(arb)` qui n'affiche que les feuilles de l'arbre binaire non vide `arb` passé en argument, ce qui correspond aux maladies possiblement induites par l'arbre de décision.

4. On souhaite maintenant afficher l'ensemble des symptômes relatifs à une maladie. On considère la fonction `symptomes(arbre, mal)` avec comme argument `arbre` un arbre de décision binaire et `mal` le nom d'une maladie.
L'appel de cette fonction sur l'arbre de décision `arb_decision` de l'introduction fournit les affichages suivants.

```
>>> symptomes(arb_decision, "anémie hémolytique")
symptômes de anémie hémolytique
splénomégalie
pas de bilirubine
conjonctive jaune
```

Pour cela, on modifie la structure précédente en ajoutant une clé `surChemin` qui sera un booléen indiquant si le nœud est sur le chemin de la maladie.
La clé `surChemin` est initialisée à `False` pour tous les nœuds.

```
arbre = {'etiquette': 'valeur' ,
         'surChemin ': False ,
         'sag': 'sous-arbre gauche' ,
         'sad': 'sous-arbre droit' }
```

Recopier et compléter les lignes 6, 8, 14 et 18 du code suivant sur votre copie.

```
01    def symptomes(arb, mal):
02        if arb['sag'] != {} :
03            symptomes(arb['sag'],mal)
04
05        if arb['sad'] != {} :
06            symptomes(.....)
07
08        if ..... :
09            arb['surChemin'] = True
10            print('symptômes de', arb['etiquette'],':')
11
12        else :
13            if arb['sad'] != {} and arb['sad']['surChemin'] :
14                print(.....)
15                arb['surChemin'] = True
16
17            if arb['sag'] != {} and arb['sag']['surChemin'] :
18                print(.....)
19                arb['surChemin'] = True
```