Exercice 4

1a

racine => “Lea”
feuilles => “Marc”, “Lea”, “Claire”, “Theo”, “Marie”, “Louis”, “Anne” et “Kevin”

1b
```
def vainqueur(arb):
 return racine(arb)
```
1c
```
def finale(arb):
 f1 = gauche(arb) f2
 = droit(arb)
     return [racine(f1), racine(f2)]
```

2a
```
from collections import
deque

def occurrences(arb, nom):
    file    =    deque([arb])
    compteur    =    0    while
    len(file)    >    0:    x    =
    file.popleft()         if
    racine(x) == nom:
            compteur = compteur + 1
        if not est_vide(gauche(x)):
        file.append(gauche(x)) if
        not est_vide(droit(x)):
            file.append(droit(x))
    return compteur
```

Il est aussi possible d’avoir une fonction récursive :

```
  def occurrences(arb, nom):
    if est_vide(arb):
      return 0
    elif racine(arb) == nom:
        res = 1
    else:
      res = 0
    return res + occurrences(gauche(arb), nom) + occurrences(droit(arb),
  nom)
```

2b

```
def a_gagne(arb, nom): return
    occurrences(arb,nom) > 1
```

3a

Les instructions proposées renvoient une valeur erronée dans le cas où le paramètre nom correspond au vainqueur du tournoi. En effet, si on considère l'arbre proposé à la question 1a, occurrences(arb, "Lea") renvoie 4 alors que Lea a joué seulement 3 matchs.

3b

```
def nombre_matchs(arb, nom):
    if vainqueur(arb)==nom :
        return occurrences(arb, nom) - 1
    else : return occurrences(arb,
        nom)
```

4

```
def liste_joueur(arb):
    if est_vide (arb):
        return []
    elif est_vide(gauche(arb)) and est_vide(droit(arb)) :
        return [racine(arb)]
    else :
        return liste_joueurs(gauche(arb)) + liste_joueurs(droit(arb))
```