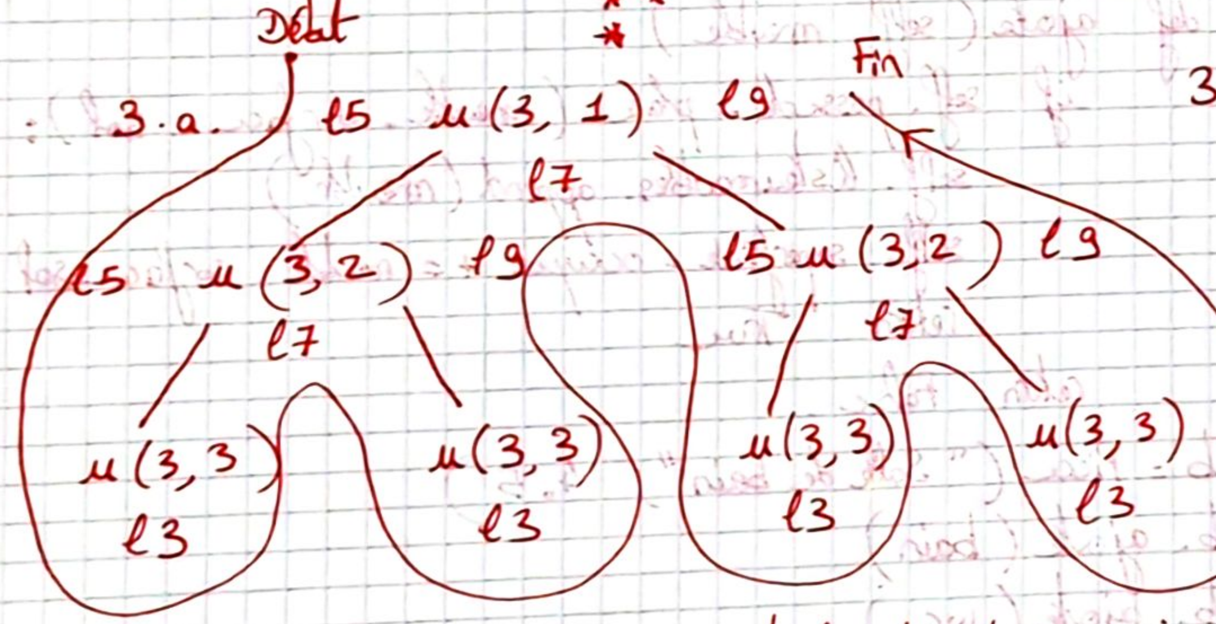
Exercice 1.

1.
```
def etoiles(n):
    if n == 0:
        return ""
    else:
        return "*" + etoiles(n-1)
```

2. f(3) affiche
```
***
**
*
```
g(3) affiche
```
*
**
***
```
h(3) affiche
```
*
**
*
***
*
**
*
```
k(3) affiche
```
***
**
*
**
***
```

3.a.



3 b. Affichage :
```
*
**
***
***
***
**
***
**
***
***
***
**
*
```

e. Comme n < i initialement et que i augmente à chaque appel récursif, le cas de base n == i ne sera jamais atteint. Python soulèvera l'erreur RecursionError.

Exercice 2.

1. `lit = Meuble("Lit Queen Size", 2 * 1.6)`

2.
```
class Piece :
    def __init__(self, n, s):
        """ Piece, str, float -> None """
        self.nom = n
        self.superficie = s
        self.liste_meubles = []
        self.superficie_occupee = 0.

    def possede_place(self, surface):
        return self.superficie - self.superficie_occupee > surface

    def ajoute(self, meuble):
        if self.possede_place(meuble.surface_sol):
            self.liste_meubles.append(meuble)
            self.superficie_occupee += meuble.surface_sol
            return True
        return False.
```

3 a.
```
sdb = Piece("Salle de bain", 4.5)
sdb.ajoute(bain)
sdb.ajoute(wc)
sdb.ajoute(lav)
couloir = Piece("Couloir", 4)
cuis = Piece("Cuisine", 3)
cuis.ajoute(ev)
cuis.ajoute(gaz)
sal = Piece("Salon", 27)
apt = Appartement([sdb, couloir, cuis, sal])
```

b.
```
def prix(self, prix_m2):
    s = 0
    for p in self.contenu:
        s += p.superficie
    return s
```

c.
```
def recherche_place(self, superficie):
    l = []
    for p in self.contenu:
        if p.possede_place(superficie):
            l.append(p)
    return l
```

4.
```
def recherche_appart(collection):
    prix_min = collection[0].prix(10000)
    apt_min = collection[0]
    for apt in collection:
        if prix_min > apt.prix(10000):
            prix_min = apt.prix(10000)
            apt_min = apt
    return apt
```