

Feuille d'exercices sur la récursivité

**EXERCICE 1 :** On appelle **fonction récursive de Ackermann** la fonction Ack qui pour deux entiers naturels  $m$  et  $n$  associe l'entier  $Ack(m,n)$  défini par :

$$Ack(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ Ack(m-1,1) & \text{si } m > 0 \text{ et } n = 0 \\ Ack(m-1, Ack(m,n-1)) & \text{si } m > 0 \text{ et } n > 0 \end{cases}$$

1) Calculer à la main :

- a)  $Ack(0,1)$     b)  $Ack(0,2)$     c)  $Ack(1,0)$     d)  $Ack(2,0)$     e)  $Ack(1,2)$

**Indications :** Vous pouvez utiliser les résultats précédents pour calculer les suivants.

2) Compléter la fonction Python suivante :

```
def ack(m, n):  
    if ..... :  
        return .....  
    elif ..... :  
        return .....  
    else:  
        return .....
```

**EXERCICE 2 :** On considère cette fonction :

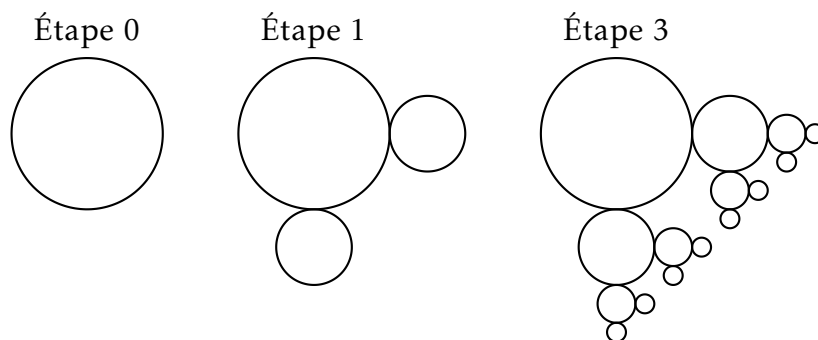
1) Déterminer le résultat de :

- a)  $mystere(5)$   
b)  $mystere(24)$   
c)  $mystere(2048)$

```
def mystere(n):  
    if n < 10:  
        return n  
    else:  
        return n%10 + mystere(n//10)
```

2) Que fait la fonction  $mystere(n)$  ?

**EXERCICE 3 :** Le schéma ci-dessous représente une figure fractale à différentes étapes.



La construction vérifie les propriétés suivantes :

- Les cercles voisins sont tangents, c'est-à-dire qu'ils se touchent en un seul point.
- Le rayon est divisé par deux lorsqu'on va vers la droite ou le bas.
- Deux cercles tangents ont des centres qui ont la même abscisse ou la même ordonnée.

Vous disposez d'une fonction `cercle(x, y, r)` qui trace un cercle de centre  $(x, y)$  et de rayon  $r$ .

Écrire une fonction récursive `fractale(x, y, r, etape)` qui trace la fractale à l'étape `etape`, avec le grand cercle centré en  $(x, y)$  et de rayon  $r$ .

```
def fractale(x, y, r, etape):
```

**EXERCICE 4 :** On considère une structure de listes chaînées avec les fonctions suivantes: `tete`, `queue` et `est_vide`. Écrire une fonction récursive `maximum1(liste)` qui renvoie la valeur maximale de la liste chaînée non vide `liste`. Attention, le cas de base n'est donc pas de la liste vide mais de la liste avec un seul élément. Vous pouvez utiliser la fonction `max(a, b)` qui renvoie le maximum des deux valeurs.

- 1) S'il n'y a qu'un seul élément, comment déterminer le maximum?
- 2) S'il y a 2 éléments ou plus, comment déterminer `maximum1(liste)` à l'aide de `tete(liste)` et `maximum1(queue(liste))`?
- 3) Compléter le code ci-dessous:

```
def maximum1(liste):  
    if ..... : # Il ne reste qu'un élément  
        return .....  
    else:  
        return .....
```

**EXERCICE 5 :** La fonction récursive `maximum2(tab, i)` renvoie la valeur maximale contenue dans le tableau `tab` à partir de l'indice `i`. On suppose que  $0 \leq i \leq \text{len}(\text{tab}) - 1$  et que `tab` est non vide.

- 1) Quelle est la plus grande valeur possible pour `i` par rapport à `tab`?
- 2) Comment déterminer `maximum2(tab, i)` si on connaît `tab[i]` et `maximum2(tab, i+1)`?
- 3) Compléter le code ci-dessous:

```
def maximum2(tab, i):  
    if ..... :  
        return .....  
    else:  
        return .....
```

**EXERCICE 6 :** On souhaite écrire une fonction récursive `binaire(n)` qui convertit un entier positif `n` en binaire. Les nombres binaires seront représentés par des chaînes de caractères avec le bit de poids fort à gauche et celui de poids faible à droite. Ainsi `binaire(4) = '100'`.

- 1) Déterminer une relation entre `binaire(n)`, `binaire(n//2)` et `str(n%2)`.
- 2) En déduire une version python de `binaire(n)`.

```
def binaire(n):
```