Afin de gagner du temps, effectuez une COPIE dans votre espace du `TP1etudiant.py` partagé sur le réseau. (n'éditez pas l'original!!!!!!!)

Exercice 1 *Dictionnaire*

On se donne le dictionnaire des coefficients CCINP 2023

```
>>> D={'info':6, 'modelisation': 7,'maths':12,'physique':13,'chimie':7, 'lettres':9, 'LV1':4}
>>> type(D)
<class 'dict'>
```

1. Proposez une commande pour :
   - ajouter une entrée LV2 affectée d'un coefficient 2
   - changer en 6 le coefficient de la modélisation
   - afficher la liste des clés de ce dictionnaire (pas des valeurs associées, à l'aide d'une boucle `for` et de la commande `D.items()`.

Exercice 2 *Récursivité*

1. Expliquez le fonctionnement de l'algorithme récursif suivant, en précisant les cas terminaux (ou de base) et en justifiant que l'algorithme s'arrête lors de l'execution sur un paramètre entier naturel.

```
def C(n:int)->int:
    '''calcule Cn récursivement'''
    if n==0:
        return(1)
    else:
        S=0
        for j in range(n):
            S=S+C(j)
        return(S)
```

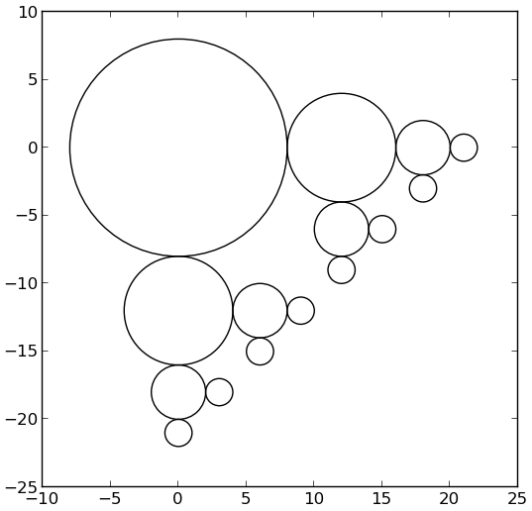
2. Expliquez pourquoi cet algorithme n'est pas optimal en temps de calcul.
3. Proposer une version mémoïsée avec un dictionnaire pour l'optimiser. on pourra compléter le code suivant :

```
dict = {}

dict[0]=...

def Cmem(n:int)->int:
    #test si C[n] est déja connu
    if n not in dict:
        res=0
        for j in range(n):
            res=...+Cmem(...)
        #Mise à jour du dictionnaire
        dict[n]=res
    else:
        res=dict[...]
    return ...

print(Cmem(3))
```

4. Visualisez l'état du dictionnaire suite à l'appel `Cmem(100)`.
5. Comparez avec `C(100)`...

Exercice 3 *Récursivité et Fractale*

On peut décrire la figure suivante de façon récursive :



La figure est formée d'un cercle et de deux copies de ce cercle ayant subies une réduction d'un facteur 2, ces deux petits cercles étant tangents extérieurement au cercle initial et tels que les lignes des centres sont parallèles aux axes du repère. Ces deux petits cercles deviennent à leur tour « cercle initial »pour poursuivre la figure.

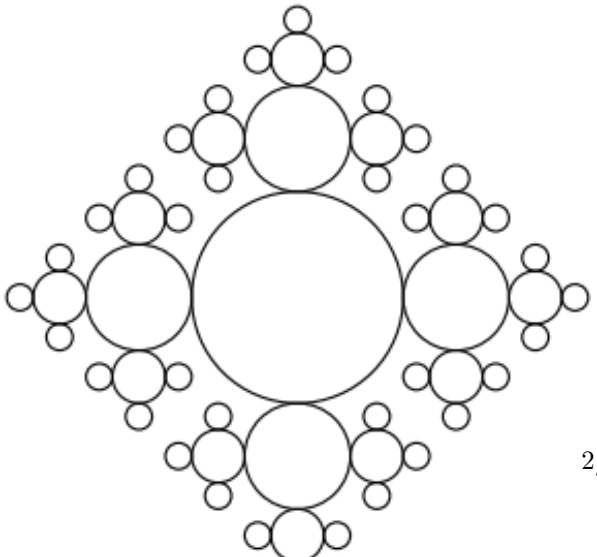
On peut traduire avec Python ce descriptif récursif de la façon suivante :

```
import pylab
F = pylab.gca() # F peut être vue comme un objet 'figure'
def cercle(x, y, r):
    """ cercle de centre (x,y) et de rayon r """
    # création du cercle:
    cir = pylab.Circle([x, y], radius = r, fill = False)
    # ajout du cercle à la figure :
    F.add_patch(cir)

def CerclesRec(x, y, r):
    """ construction récursive de la figure """
    cercle(x, y, r)
    if r > 1:
        CerclesRec(x+3*r/2, y, r/2)
        CerclesRec(x, y-3*r/2, r/2)

# appel de la fonction CerclesRec
CerclesRec(0, 0, 8)
# pour placer toute la figure dans un repère orthonormé :
pylab.axis('scaled')
# affichage de la figure :
pylab.show()
```

1. Qu'est ce qui garantit que cette fonction ne s'appellera qu'un nombre fini de fois ?

2. Dans l'appel initial, si l'on change `CerclesRec(0, 0, 8)` en `CerclesRec(0, 0, 64)` , qu'obtiendra-t-on ?

3. Modifier le programme python précédent pour obtenir la figure ci-contre :



On pourra utiliser un paramètre supplémentaire pour définir la fonction récursive (abscisse du centre, ordonnée du centre, rayon du cercle, position du voisin) *où position sera l'une des chaînes de caractères : haut, bas, droite, gauche*

Exercice 4 *Récursivité et Dictionnaire*

On considère la suite récurrente (u_n) définie par :

$$\forall j \in \{0,1,2\},\ u_j = j+1$$

$$\forall n \in \mathbb{N}, u_{n+3} = \frac{u_n \times u_{n+1}}{u_{n+2}}$$

Proposer une procédure `U(n)` en langage `Python` récursive utilisant un dictionnaire pour calculer efficacement le $n$ème terme de la suite, on pourra compléter le code suivant

```
...= {0:1,1:2,2:3}


def U(n:int)->float:
    #test si U[n] est déja connu
    if n .........:
        res=Dc[n]
    else:
        res=...
        #mise à jour du dictionnaire
        ...=res
    return res
```

Exercice 5 *Algorithme mystère*

Soit la fonction $mystere(t,k)$ où t est un tableau d'entiers non vide et k verifiant $0 \leq k < len(t)$.

```
def mystere(t ,k):
   if k==len(t)-1:
      return True
   if t[k]>t[k+1]:
       return False
   return mystere(t,k+1)
```

1. Soit $t = [6,9,4,8,12]$. Que retourne $mystere(t,2)$ (Donner la liste des appels récursifs) ?
2. Même question pour $mystere(t,0)$
3. Que fait la fonction $mystere$ dans le cas général ?
4. Quel est le nombre maximum d'appels récursifs (en fonction de n et k) de la fonction $mystere(t,k)$ si le tableau t est de longueur n ?