```
## exo 1

def q1(n):
    if n==0:
        return 2
    else:
        return 1/(q1(n-1)+2)

def q2(n):
    if n==1:
        return 3
    else:
        return (q2(n-1)-2*(n-1))/3

def q3(n):
    if n==1:
        return 2
    else:
        a = q3(n-1)
        return a+a**(1/2)  # attention à ne pas écrire q3(n-1) + q3(n-1)**(1/2) qui renvoie le
même résultat mais avec beaucoup trop d'appels récursifs !

def q4(n):
    if n==0 or n==1:
        return 1
    else:
        return q4(n-1) + q4(n-2)

# il y a beaucoup plus de calculs inutiles dans cette version récursive que dans la version
avec une boucle for

def q4bis(n):
    u,v = 1,1
    for k in range(n-1):
        u,v = v, u+v
    return v

def q5(n):
    if n<2:
        return 0
    else :
        return q5(n-1) + n**(1/2)/(1+n)

def q6(n):
    if n<0:
        return 1
    else :
        return q6(n-1)*((n+1)**(1/2)+1)

## exo2

# question 1
def u(n):
    if n==0:
        return 2
    else:
        if n%2==0:
            a = u(n//2)
            return a+a**2-n
        else:
            a = u(n-1)
            return (a+1)//(2*a+3)
##


# question 2
def v(n):
    if n==0:
        return 0
    else:
        if n%2==0:
            a = v(n//2)
            return 4*a
        else:
            a = v((n-1)//2)
            return a+2*(n-1)+1
```

```
## exo 3
# q1 : v_3(18)=2 car 18=2*3^2, v_3(12)=1 car 12=4*3^1, v_3(11)=0 car 11 n'est pas divisible par
3

# q2 : v_p(n)=0 lorsque n n'est pas divisible par p c'est-à-dire en Python lorsque n%p != 0

# q3 : dans ce cas, v_p(n)=1+v_p(n/p)

# q4
def valuation(p,n):
    if n%p != 0:
        return 0
    else :
        return 1+valuation(p,n//p)

# on teste la fonction sur des exemples

## exo 4
# q1 : (k parmi n) = (k parmi n-1) + (k-1 parmi n-1)

# q2 : dans ce cas (a parmi b) = 0 et la formule est valable (sur les bords du triangle de
Pascal, on fait comme si à l'extérieur tous les coefficients valent 0)

# q3 : on initialise au cas n=0 et aux cas extérieurs au triangle puis on utilise la formule de
Pascal

def binom(k,n):
    if n==0 and k==0:
        return 1
    if k>n or k<0:
        return 0
    else :
        return binom(k,n-1)+binom(k-1,n-1)

# q4, on utilise une variable globale N (càd on n'est pas dans une fonction)
N = 5
for n in range(N+1):
    for k in range(n+1):
        print(binom(k,n), end='') # pas de retour à la ligne quand on change de k
    print() # pour faire le retour à la ligne quand on change de n

# on peut aussi s'amuser à mettre des espaces pour rendre tout cela plus lisible ou pour écrire
le triangle de Pascal ''en pyramide''.


## exo 5


##
import matplotlib.pyplot as plt
axe = plt.gca()

def cercle(x,y,r):
    c = plt.Circle((x, y), r, color='b',fill=False)
    axe.add_patch(c)

def montre():
    plt.axis('scaled')
    plt.show()

# q1

def recu1(n,x,y,r):
    cercle(x,y,r)
    if n>0:
        recu1(n-1,x+r+r/2,y,r/2)
        recu1(n-1,x,y+r+r/2,r/2)

# pour tester la q1
recu1(4,0,0,1) # pour avoir le premier dessin
montre()

##

import matplotlib.pyplot as plt
```

```
axe = plt.gca()

def cercle(x,y,r):
    c = plt.Circle((x, y), r, color='b',fill=False)
    axe.add_patch(c)

def montre():
    plt.axis('scaled')
    plt.show()

# q2
def recu2(n,x,y,r,dir):
    new_r = r/2.2
    if dir == 'all':
        recu2(n,x,y,r,'n')
        recu2(n-1,x,y-r-new_r,new_r,'s')
    if n==0 :
        cercle(x,y,r)
    else :
        cercle(x,y,r)
        if dir=='o':
            recu2(n-1,x-r-new_r,y,new_r,'o')
            recu2(n-1,x,y+r+new_r,new_r,'n')
            recu2(n-1,x,y-r-new_r,new_r,'s')
        if dir=='s':
            recu2(n-1,x+r+new_r,y,new_r,'e')
            recu2(n-1,x,y-r-new_r,new_r,'s')
            recu2(n-1,x-r-new_r,y,new_r,'o')
        if dir=='n':
            recu2(n-1,x+r+new_r,y,new_r,'e')
            recu2(n-1,x,y+r+new_r,new_r,'n')
            recu2(n-1,x-r-new_r,y,new_r,'o')
        if dir=='e':
            recu2(n-1,x,y-r-new_r,new_r,'s')
            recu2(n-1,x,y+r+new_r,new_r,'n')
            recu2(n-1,x+r+new_r,y,new_r,'e')

# test pour la q2
recu2(4,0,0,1,'all') # pour avoir le deuxième dessin
montre()
```