```
# ================================= #
# MP/PC-PC*/PSI*                    #
# Concours blanc - 10 janvier 2018  #
# EXERCICE I                        #
# LA FONCTION D'ACKERMANN           #
# Une folie récursive ! :)          #
# ================================= #

## Importations
from sys import setrecursionlimit

## Initialisations
setrecursionlimit(10000)

## Question 1. Fonction AP
def AP(m,n):
    # On vérifie d'abord que les deux arguments sont bien des entiers...
    if type(m) != int or type(n) != int:
        raise TypeError("Les deux arguments doivent être des entiers !")
    # ... et ensuite qu'ils sont positifs !
    elif m < 0 or n < 0:
        raise ValueError("Les arguments doivent être des entiers naturels")
    # C'est bien le cas...
    else:
        if m == 0:
            return(n + 1)
        else:
            if n == 0:
                return(AP(m - 1,1))
            else:
                return(AP(m - 1,AP(m,n - 1)))

## Question 4. Fonction KnuthPI
# Un premier code en prenant directement la définition...
def KnuthPI(a,b,n):
    if type(a) != int or type(b) != int or type(n) != int:
        raise TypeError("Les trois arguments doivent être des entiers !")
    elif a < 0 or b < 0:
        raise ValueError("Les deux premiers arguments doivente être des
entiers naturels !")
    elif n < 1:
        raise ValueError("Le troisième argument doit être un entier naturel
non nul !")
    else:
        if n == 1:
            return(a**b)
        elif b == 0:
            return(1)
        else:
            return(KnuthPI(a,KnuthPI(a,b-1,n),n-1))

# Un deuxième code en étant moins "violent" au niveau de la récurrence...
def KnuthPI2(a,b,n):
    if type(a) != int or type(b) != int or type(n) != int:
        raise TypeError("Les trois arguments doivent être des entiers !")
    elif a < 0 or b < 0:
        raise ValueError("Les deux premiers arguments doivente être des
entiers naturels !")
    elif n < 1:
        raise ValueError("Le troisième argument doit être un entier naturel
```

```
non nul !")
    else:
        if n == 1:
            return(a**b)
        else:
            R = 1
            for k in range(b):
                R = KnuthPI2(a,R,n-1)
            return(R)

## Question 5.
def AP2(m,n):
    if type(m) != int or type(n) != int:
        raise TypeError("Les deux arguments doivent être des entiers !")
    elif m < 0 or n < 0:
        raise ValueError("Les arguments doivente être des entiers naturels
!")
    else:
        if m == 0:
            return(n + 1)
        elif m == 1:
            return(n + 2)
        elif m == 2:
            return(2*n + 3)
        else:
            return(KnuthPI2(2,n+3,m-2)-3)

## Pour tester... avec beaucoup de modération !!! Prudence dès que m >= 4 !
print("Test de la fonction AP (à tester avec de très petites valeurs !)")
m = int(input("Veuillez saisir m : "))
n = int(input("Veuillez saisir n : "))
print(AP(m,n))

print("Test de la seconde fonction")
m = int(input("Veuillez saisir m : "))
n = int(input("Veuillez saisir n : "))
print(AP2(m,n))
```