```
from tkinter import *
#Le probleme des N reines
#(Solution par recursivite)
#(Recherche en profondeur d'abord)

#variables globales:
echiquier=[] #tableau NxN
pos_reine=[] #Les positions des N reines
sol=[]
#fonctions & procedures

#Precedure d'initialisation :
#- On fixe la taille de l'echiquier (tableau NxN) et on l'initialise
par 0
#- On fixe la taille du tableau contenant les positions des N reines
et on l'initialise par 0

def init_echiquier(n):
    for i in range(0,n):
        pos_reine.append(0)
        li=[]
        for k in range(0,n):
            li.append(0)
        echiquier.append(li)


#Procedure d'affichage de toutes les solutions possibles

def affiche():
    for ligne in echiquier:
        for case in ligne:
            print(case,"",end="")
        print("")
    print("")

#Procedure pour 'poser les reines' sur l'echiquier
#-On remplit toutes les cases vides de l'echiquier par 0, et les
cases contenant les reines par 1
#-et on affiche le resultat

def poser(n):
    for i in range(0,n):
        for k in range(0,n):
            echiquier[i][k]=0
    for i in range(0,n):
        #echiquier[i][pos_reine[i]]=1
        echiquier[pos_reine[i]][i]=1
    affiche()


#Fonction de teste si la case est valide:
#Si la case est valide on retourne 1 sinon on retourne 0
#-Teste si la ligne est vide
#-Teste si la reine n'est pas a la portee d'une autre reine en
diagonale
```

```
#-On pose les reines en se deplacant a droite sur l'echiquier
suivant les colonnes,
#donc on ne fait pas de test sur les colonnes.

def valide(li ,co):
    for i in range(0,co):
        if pos_reine[i]==li or abs(pos_reine[i]-li)==abs(i-co):
            return 0
    return 1


#- Une fonction recursive qui fait une recherche en profondeur
d'abord :
#- continue a developper les resultats trouves en profondeur tant
que valide est 'true', en se deplacant a droite suivant les
colonnes.
#- S'il n'est plus possible de poser une reine sur une nouvelle
colonne, la fonction recule a la colonne precedante et change la
position de la reine qui s'y trouve

def cherche_solution(r,n):
    global k,x,t,sol  #pour compter les solutions
    if(r==n): #si on arrive a poser toutes les reines on affiche le
resultat qui represente l'une des configurations possibles
        sol=sol+pos_reine
        print(pos_reine)
        x=int(k.get())+1
        k.set(x)
    else :
#pour une colonne donne, on teste si l'une des ligne est valide pour
y poser la reine,
#si aucune ligne n'est valide, on retourne a la colonne precedante
et change la position de la reine qui s'y trouve
      for i in range(0,n): #la colonne est fixe, et on teste pour
toutes les lignes
            if(valide(i,r)): #teste si la case est valide
                pos_reine[r]=i #si la case (colonne=r, ligne=i) est
valide on y pose une reine
                cherche_solution(r+1,n) #on passe a la colonne
suivante

def calculer(n):
    global k,clic,pos_reine,sol
    sol=[]
    clic.set(0)
    pos_reine=[]
    k.set(0)
    init_echiquier(n)
    cherche_solution(0,n)


def voir(n):
   global clic,k
   cli=int(clic.get())-1
   d=600//n
   for i in range(n):
```

```
      r=list(range(n))
      for j in range(n):
         r[i]=list(range(n))
         lo=i*d
         ha=j*d

r[i][j]=c.create_rectangle(lo,ha,lo+d,ha+d,outline='black',fill='whi
te')
   for i in range (n*cli,n*cli+n):
      c.create_oval(sol[i]*d+1,(i-n*cli)*d+1,sol[i]*d+d-1,(i-
n*cli)*d+d-1,fill='black')
   if cli+1<int(k.get()):
         cli=cli+1
         clic.set(cli+1)



#debut programme :
x=0
sol=[]
#cherche_solution(0)
f=Tk()
clic=StringVar()
clic.set(1)
k=StringVar()
k.set(0)
c = Canvas(f,bg='white',height=600,width=600)
c.pack(side=LEFT)
spin=Spinbox(f, values=(4,5,6,7,8,9,10,11), width=4)
spin.config( font="sans 12", justify="center")
spin.pack()
b=Button(f,text="calculer",command=lambda:calculer(int(spin.get())))
b.pack()
ll=Label(f,text="nombre de solutions")
ll.pack()
l=Label(f,textvariable=k)
l.pack()
Label(f).pack()
bb=Button(f,text="voir des
solutions",command=lambda:voir(int(spin.get())))
bb.pack()
lll=Label(f,textvariable=clic)
lll.pack()
f.mainloop()
print(k.get())
print(sol)
```