```
from math import *

def prime(atester):
  pastrouve = True
  k = 2
  if (atester == 1): return False
  if (atester == 2): return True
  if (atester == 3): return True
  if (atester == 5): return True
  if (atester == 7): return True
  while (pastrouve):
    if ((k * k) > atester):
        return True
    else:
        if ((atester % k) == 0):
            return False
        else: k=k+1

for n in range(14,102,2):
  pix=0 ; pi2x = 0
  for compteprem in range(2,n+1,1):
    if (prime(compteprem)):
      if (compteprem <= n//2): pix=pix+1
      pi2x=pi2x+1
  xa=0 ; xb=0 ; xc=0 ; xd=0
  ya=0 ; yc=0
  za=0 ; zc=0
  zaprec = za ; zcprec = zc ; yaprec=ya ; ycprec=yc
  for x in range(3,n//2+1,2):
    if (prime(x)):
      if (prime(n-x)): xa=xa+1
      else: xc=xc+1
    else:
      if (prime(n-x)): xb=xb+1
      else: xd=xd+1
  for x in range(6,((n+4)//2)+1,2):
    if (prime(x-3)): za=za+1
    else: zc=zc+1
  if ((n//2) % 2 == 0): debuthaut=((n+4)//2)+2
  else: debuthaut = ((n+4)//2)+1
  for x in range(debuthaut, n+1, 2):
     if (prime(x-3)): ya=ya+1
     else: yc=yc+1
  pix=0 ; pi2x = 0
  for compteprem in range(2, n+1, 1):
     if (prime(compteprem)):
        if (compteprem <= n//2): pix=pix+1
        pi2x=pi2x+1
  print("\n",n,"  :  ")
  print("Xa ",xa, "Xb ",xb, "Xc ",xc, "Xd ",xd)
  print("Ya ",ya, "Yc ",yc)
  print("Za ",za, "Zc ",zc)
  print("(n-4)/4 ",(n-4)//4,"(n-2)/4", (n-2)//4)
  if ((n%4) == 0): print("bool_4k+2 0 ")
  else:
    print("bool_4k+2 1 ")
    if (prime(n//2)) :
       print("bool_2p 1 ")
       print("bool_2c-imp 0")
    else:
       print("bool_2p 0 ")
       print("bool_2c-imp 1")
  print("n/4 ",n//4," pi(n) ",pi2x," pi(n/2) ",pix)
  if (prime(n-1)): print("n-1 premier ")
```

```
  else: print("n-1 composé ")
  if (prime(n+1)): print("n+1 premier ")
  else: print("n+1 composé ")
  if (prime(n//2)): print("n/2 premier ")
  else: print("n/2 composé ")
  if (prime((n+2)//2)): print("(n+2)/2 premier")
  else: print("(n+2)/2 composé")
  print("d_za ",za-pix, "d_zc ",zc-n//4+pix)
  print("d_ya ",ya-pi2x+pix,"d_yc ",yc-n//4+pi2x-pix)
  print("d_zc-za ",zc-za-n//4+2*pix,"d_za-ya ",za-ya-2*pix+pi2x,"d_yc-zc ",yc-
zc-2*pix+pi2x)
  print("d_zc-ya ",zc-ya-n//4+pi2x,"d_yc-za ",yc-za-n//4+pi2x,"d_xd-xa ",xd-xa-
n//4+pi2x)
##############################################################################
# valeurs des variables
##############################################################################
#
# xa est le nombre de décompositions de n comme somme de 2 impairs premiers
# p1 et p2
# n=p1+p2 avec 3 <= p1 <= n/2 et n/2 <= p2 <= n-3 (ces bornes des intervalles
# d'appartenance de p1 et p2 sont identiques dans les définitions de xb, xc
# et xd ci-dessous)
# xb est le nombre de décompositions de n comme somme d'un impair premier
# p1 et d'un impair composé p2
# xc est le nombre de décompositions de n comme somme d' un impair composé
# p1 et d'un impair premier p2
# xd est le nombre de décompositions de n comme somme de 2 impairs composés
# p1 et p2
# ya est le nombre de décompositions de n de la forme 3+p avec p premier >= n/2
# yc est le nombre de décompositions de n de la forme 3+c avec c composé >= n/2
# za est le nombre de décompositions de n de la forme 3+p avec p premier < n/2
# zc est le nombre de décompositions de n de la forme 3+c avec p premier < n/2
# bool_4k+2 vaut 1 si n est de la forme 4k+2 et 0 sinon.
# bool_2p vaut 1 si n est le double d'un nombre premier et 0 sinon.
# bool_2c_imp vaut 1 si n est le double d'un nombre impair composé et 0 sinon.
#
##############################################################################
# invariants
##############################################################################
#
# ya=xa+xb
# yc=xc+xd
# ya+yc=xa+xb+xc+xd=(n-2)//4
# za+zc=(n-4)//4
# xa+xc=za+delta_za+bool_2p
# xb+xd=zc+bool_2c_imp
# zc-ya=xd-xa-bool_2c_imp
# zc-ya=yc-za-bool_4k+2
# za = pi(n/2)+delta_za avec delta_za=-2 si n est le double d'un nombre
# premier et -1 sinon.
# zc=n/4-pi(n/2)+delta_zc avec delta_zc=1 si n est le double d'un nombre
# premier et 0 sinon
# ya=pi(n)-pi(/2)+delta_ya avec delta_ya=-1, 0 ou 1 suivant les caractères
# de primalité de n-1 et n/2
# (se reporter à https://hal.archives-ouvertes.fr/hal-01109052)
# yc=n/4-pi(n)+pi(n/2)+delta_yc avec delta_yc=-1, 0 ou 1
# (se reporter au document référencé ci-dessus)
# (n < 240) ou (ya < za < zc < yc) est toujours vraie
# (n < 244) ou (yc+zc < xd) est toujours vraie
#
##############################################################################
```