```python
'''
CHP 1 : Structures de données avancées
Série : TP 2 les piles
Groupe : Ipein/SM1/GB
Date : 18/09/2023
'''
# Pile
def creer_pile():
   return list()#[]

def pile_vide(p): return len(p)==0 #p==[]

def sommet(p):
   if not pile_vide(p):
       return p[-1]
   else:
       raise Exception("Pile vide")
# prog ppl
p = creer_pile()
try:
   s = sommet(p)
except:
   print("Pas sommet: pile vide")
else:
   print("sommet:",s)

def taille(p): return len(p)
def empiler(p,x): p.append(x)
def depiler(p):
   if not pile_vide(p):
       return p.pop()
   else:
       raise Exception("Pile vide")

# Ex 1
def conversion(n):
   p= creer_pile()
   if n == 0 : return '0b0'
   while n != 0: # ! not = equal !=
       n,r = divmod(n,2)
       empiler(p,r)
   result = '0b'
   while not pile_vide(p):
       result += str(depiler(p))

   return result

# Ex 2
#eval
def verif_parentheses(expr):
   p = creer_pile()
   L = []
   for i in range(len(expr)):
       if expr[i] == '(':
           empiler(p,i)
       elif expr[i] == ')':
           if pile_vide(p): return False
           L.append((depiler(p),i))
   if pile_vide(p): return L
   return False

def calc_expr_arith(expr):
   p = creer_pile()
   for c in expr :
       if c.isdigit() :
           empiler(p,int(c))
       else:
           n1, n2 = depiler(p), depiler(p)
           empiler(p,eval(str(n2)+c+str(n1)))

   return depiler(p)

# Ex 3
def permut_circ(p,n):
   p1 = creer_pile()
   p2 = creer_pile()
   for i in range(n):
       empiler(p1,depiler(p))

   for i in range(taille(p)-n):
       empiler(p2,depiler(p))

   while taille(p1)>0:
       empiler(p,depiler(p1))

   while not pile_vide(p2):
       empiler(p,depiler(p2))

# Ex4
def somme(p):
   if taille(p)==0: return 0
   else:
       s = depiler(p)
       if type(s)==int:
           return s + somme(p)
       else:
           return somme(s)+somme(p)
```