```python
def creer_pile():
   return []

def pile_vide(p):
   return len(p)==0

def sommet(p):
   if not pile_vide(p):
       return p[-1]

p = creer_pile()
if sommet(p) :
   s = 2 + sommet(p)

def taille(p):
   return len(p)

def empiler(p,x):
   p.append(x)

x = 3
empiler(p,x)

def depiler(p):
   if not pile_vide(p):
       r = p.pop()
       return r
   else:
       raise Exception("Pile vide")

def depiler1(p):
   assert len(p)>0
   return p.pop()

try :
   s = depiler(p)
except :
   print("Pile vide")
else:
   r = s + 2

# ex1
def conversion(n):
   if n==0 : return '0b0'
   p = creer_pile()
   while n!= 0:
       n,r = n//2,n%2
       empiler(p,r)
   result = '0b'
   while taille(p)>0:
       result += str(depiler(p))
   return result

# ex 2
def verif_parenthese(ch):
   p = creer_pile()
   L = []
   for i in range(len(ch)):
       if ch[i] == '(': empiler(p,i)
       elif ch[i] == ')':
           if pile_vide(p): return False
           L += [(depiler(p),i)]

   return L if pile_vide(p) else False


def cal_expr_arith(expr):
   '''
   expr : un expression arithmétique postfixée
   '''
   p = creer_pile()
   for c in expr:
       if c.isdigit():
           empiler(p,int(c))
       else:
           n1,n2 = depiler(p),depiler(p)
           if c=='+': r = n2 + n1
           elif c=='-': r = n2 - n1
           elif c== '/': r = n2 / n1
           elif c== '*': r = n2 * n1
           empiler(p,r)
   return sommet(p)
```

```
# Ex 3
def premut_circ(p,n):
   # repeter n fois
   for j in range(n):
       # mettre le sommet au fond de la pile
       s = depiler(p)
       # vider la pile p dans p1
       p1 = creer_pile()
       while not pile_vide(p):
           x = depiler(p)
           empiler(p1,x)
       #mettre s dans p (p est vide)
       empiler(p,s)
       t = taille(p1)
       for i in range(t):
           empiler(p,depiler(p1))

# ex 4
def somme(p):
   if pile_vide(p): return 0
   else:
       s = depiler(p)
       return somme(p) + s if type(s)==int else somme(s)
       """
       if type(s) == int:
           return s +  somme(p)
       else:
           return somme(s) + somme(p)
       """
```