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

p = creer_pile()

def pile_vide(p):
   #return True if len(p)==0 else False
   return len(p)==0

vide = pile_vide(p)

def sommet(p):
   if not pile_vide(p):
       return p[-1] #p[len(p)-1]
   else:
       #print("pile vide")
       raise Exception("pile vide")

s = sommet(p)

def taille(p):
   return len(p)

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

def depiler(p):
   return p.pop()#par défaut index=-1

#
try:
   s = depiler(p)
except :
   print("pile vide")
else:
   p1 = creer_pile ()
   empiler(p1,s)


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

# Ex 2
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+=[(depiler(p),i)]


   return L if pile_vide(p) else False

# https://anis-saied.github.io/ipein

# q3
def calcul_expr_arith(ch):
   p = creer_pile()
   for c in ch:
       if c.isdigit():
           empiler(p,c)
       else:
           n1,n2 = depiler(p), depiler(p)
           empiler(eval(n2+c+n1))
   return sommet(p)

# ex3
def permut_circ(p,n):
   p1 = creer_pile()
   for i in range(n):
       x = depiler(p)
```

```
        #vider la pile p dans p1
        for j in range(taille(p)):
            empiler(p1, depiler(p))

        empiler(p,x)

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



# ex4
```