LISTES, PILES, FILES – EXERCICES CORRECTION

Les piles

Exercice 1 (débranché)

P1 :

L
O
V
E

P2 :

L
A
C
E

Exercice 2 (débranché)

P1 :

1
0

P2 :

9
25
49
81

P3 :

1
9
25
49

Exercice 3 : une première implémentation avec des tuples

```
def creer_p():
    return ()

def estVide_p(P):
    return P==()

def empiler(P,e):
    return (e,P)

def depiler(P):
    assert not estVide_p(P) , 'attention, la pile est vide!!'
    return P[1]

def sommet(P):
    assert not estVide_p(P) , 'attention, la pile est vide!!'
    return P[0]

def base(P):
    assert not estVide_p(P) , 'attention, la pile est vide!!'
    return P[1]

def longueur_p(P):
    cpt=0
    p=P
    while p!=():
        p=base(p)
        cpt+=1
    return cpt
```

```
def dernier_p(P):
    assert not estVide_p(P) ,'attention, la pile est vide!!'
    p=P
    while p[1]!=():
        p=base(p)
    return sommet(p)

def inserer_p(P,e,i):
    assert longueur_p(P)>=i,'attention, pile trop petite'
    p1=creer_p()
    p=P
    cpt=0
    while cpt!=i-1:
        p1=empiler(p1,sommet(p))
        p=base(p)
        cpt+=1
    p=empiler(p,e)
    for k in range(i-1):
        p=empiler(p,sommet(p1))
        p1=base(p1)
    return p
```

Exercice 4 : une deuxième implémentation avec des tableaux

```
def creer_p():
    return [0] + [None]*10

def estVide_p(P):
    return P[0]==0

def empiler(P,e):
    assert P[0]!=10 ,"Attention, pile pleine"
    P[0]+=1
    P[P[0]]=e

def depiler(P):
    assert not estVide_p(P),'Attention, pile vide!'
    P[P[0]]=None
    P[0]-=1

def sommet(P):
    assert not estVide_p(P),'Attention, pile vide!'
    return P[P[0]]
```

```
def base(P):
    assert not estVide_p(P),'Attention, pile vide!'
    return [P[0]-1]+P[1:P[0]]+P[P[0]+1:11]+[None]
# OU BIEN
def base(P):
    P1=P
    depiler(P1)
    return P1

def longueur_p(P):
    return P[0]

def dernier_p(P):
    assert not estVide_p(P),'Attention, pile vide!'
    return P[1]

def inserer_p(P,e,i):
    assert longueur_p(P)>=i,'attention, pile trop petite'
    assert P[0]!=10,"Attention, pile pleine"
    for k in range(P[0],i-1,-1):
        P[k+1]=P[k]
    P[i]=e
    P[0]+=1

def ieme_p(P,i):
    assert P[0]>=i,'Attention, pile trop petite!'
    return P[i]
```

Les files

Exercice 5 (questions de cours)

1. a. L'acronyme "FIFO" signifie « First In, First Out »
   b. Cet acronyme correspond au système de fonctionnement d'une file.
2. a. L'acronyme "LIFO" signifie « Last In, First Out »
   b. Cet acronyme correspond au système de fonctionnement d'une pile.

Exercice 6 (débranché)

F1 :		L	U	T	T	E	
F2 :		S	A	L	U	T	

Exercice 7 : une première implémentation avec un tableau

```
def creer_f():
    return [0] + [None]*10

def estVide_f(F):
    return F[0]==0

def debut(F):
    assert not estVide_f(F),'Attention, file vide!'
    return F[F[0]]

def suite(F):
    assert not estVide_f(F),'Attention, file vide!'
    return [F[0]-1]+F[1:F[0]]+F[F[0]+1:11]+[None]

def enfiler(F,e):
    assert F[0]!=10 ,"Attention, file pleine"
    F[0]+=1
    F[F[0]]=e

def defiler(F):
    assert not estVide_f(F),'Attention, file vide!'
    for i in range(1,F[0]):
        F[i]=F[i+1]
    F[F[0]]=None
    F[0]-=1
```

```
def longueur_f(F):
    return F[0]

def dernier_f(F):
    assert not estVide_f(F),'Attention, file vide!'
    return F[F[0]]

def inserer_f(F,e,i):
    assert longueur_f(F)>=i,'Attention, file trop petite'
    assert F[0]!=10,"Attention, file pleine"
    for k in range(F[0],i-1,-1):
        F[k+1]=F[k]
    F[i]=e
    F[0]+=1

def ieme_f(F,i):
    assert F[0]>=i,'Attention, file trop petite!'
    return F[i]
```

Exercice 8 : une deuxième implémentation avec un tableau circulaire

```
def creer_f():
    return ([None]*10,0,0,0)

def estVide_f(F):
    return F[3]==0

def debut(F):
    assert not estVide_f(F),"Attention, votre file est vide !"
    T,d,f,t=F
    return T[d]

def suite(F):
    assert not estVide_f(F),"Attention, votre file est vide !"
    T,d,f,t=F
    if t==1: #cas ou il n'y a qu'un élément dans la file : la suite est vide
        return []
    elif d==9: #cas où le début de la file est le dernier élement du tableau
        return T[0:f]
    elif d>=f: #cas ou la file commence à la fin du tableau et finit au début du tableau
        return T[d+1:10]+T[0:f]
    else :
        return T[d+1:f]
```

```
def enfiler(F,e):
    T,d,f,t=F
    assert not t==10 ,"Attention, votre file est pleine !"
    T[f]=e
    if f==9:
        f=0
    else :
        f+=1
    t+=1
    return (T,d,f,t)


def defiler(F):
    assert not estVide_f(F),"Attention, votre file est vide !"
    T,d,f,t=F
    T[d]=None
    t-=1
    if d==9:
        d=0
    else :
        d+=1
    return (T,d,f,t)
```