TD 2

Exercices d'applications

Savoirs et compétences :

- ❒ *Alg – C16 : Piles - Algorithmes de manipulation : fonctions «push» et «pop».*

Dans tous ces exrcices on utilisera uniquement les primitives `creer_pile`, `est_vide`, `est_pleine`, `empiler`, `depiler` et `taille_pile`.

Exercice 1

Écrire une fonction concaténant deux piles.
Cette fonction prendra en argument deux piles et placera la première "sur" la deuxième. Elle ne renverra rien, mais modifiera les deux piles : à la fin, la première pile sera vide et la seconde aura été augmentée de la première. Pour cela, on supposera que la seconde pile est assez grande pour recevoir toute la première.

Corrigé 1

```
def concat (P1,P2) :
    if est_vide(P1) :
        return None
    else :
        x = depiler(P1)
        concat(P1,P2)
        empiler(P2,x)
```

Exercice 2

Écrire une fonction renvoyant la plus grande somme possible de trois éléments consécutifs d'une pile d'entiers naturels. Si la pile a deux éléments ou moins, on renverra 0. On prendra garde de recopier la pile passée en argument afin de ne pas la modifier.

Corrigé 2

```
def somme3 (P) :
    R = P.copy()
    if est_vide(R) :
        return 0
    else :
        x = depiler(R)
        s = somme3(R)
        for i in range(2) :
        if est_vide(R) :
            return s
        else :
            x += depiler(R)
        return max(x,s)
```

Exercice 3

On se donne deux piles P_1 et P_2, et on se propose de fusionner ces deux piles en une seule pile P de manière à respecter les règles suivantes :

1. Si $P_1 = [a_1, a_2, \cdots, a_{n-1}, a_n]$ et $P_2 = [b_1, b_2, \cdots, b_{p-1}, b_p]$, alors $P = [\cdots, b_{p-1}, a_{n-1}, b_p, a_n]$.
2. Si l'une des pilees est vide, alors on n'a plus recours qu'à l'autre, jusqu'à ce que celle-ci aussi soit vide.

Exemples : $P_1 = (a_1, a_2, a_3)$ et $P_2 = (b_1, b_2, b_3, b_4, b_5)$, alors $P = [b_1, b_2, a_1, b_3, a_2, b_4, a_3, b_5]$.

Écrire une fonction python réalisant cela. On prendra garde de recopier les piles passées en argument afin de ne pas la modifier.
Afin d'être sûr que la nouvelle pile est assez grande pour contenir les deux premières, on pourra commencer par écrire une fonction recopiant une pile dans une autre pile de taille donnée.

Corrigé 3

```
def copie (P,n) :
    """ recopie la pile P dans une pile de taille n.
    préconditon : n > taille_pile(P) """
    l = taille_pile(P)
    R = P.copy()
    assert n>l
    if est_vide(R) :
        return creer_pile(n)
    else :
        x = depiler(R)
        S=copie(R,n)
        empiler(S,x)
        return S

def fusion (P1,P2) :
    n = taille_pile(P1)+taille_pile(P2)
    R1 = P1.copy()
    R2 = P2.copy()
    if est_vide(R1) :
        return copie(R2,n)
    if est_vide(R2) :
        return copie(R1,n)
    else :
        e1 = depiler(R1)
        e2 = depiler(R2)
        R3 = fusion(R1,R2)
        empiler(R3,e1)
        empiler(R3,e2)
        return R3
```