Devoir Surveillé : Epreuve pratique	Thème : Epreuve Pratique BAC
	TYPE BAC

Exercice n°1 :

Exercice n°1 :

Écrire une fonction `recherche` qui prend en paramètre un tableau de nombres entiers `tab`, et qui renvoie la liste (éventuellement vide) des couples d'entiers consécutifs successifs qu'il peut y avoir dans `tab`.

Exemples :

```
>>> recherche([1, 4, 3, 5])
[]
>>> recherche([1, 4, 5, 3])
[(4, 5)]
>>> recherche([7, 1, 2, 5, 3, 4])
[(1, 2), (3, 4)]
>>> recherche([5, 1, 2, 3, 8, -5, -4, 7])
[(1, 2), (2, 3), (-5, -4)]
```

Exercice n°2 :

Exercice n°2 :

On veut réaliser une implémentation objet d’une file en utilisant deux piles.

Vous utiliserez l’implémentation suivante d’une pile pour travailler

```
class Pile:
    def __init__(self):
        self.contenu = []

    def empiler(self, e):
        self.contenu.append(e)

    def depiler(self):
        assert self.taille != 0, "on ne peut pas dépiler une pi
        self.contenu.pop()

    def sommet(self):
        assert self.taille != 0, "une pile vide n’a pas de somm
        return self.contenu[-1]

    def taille(self):
        return len(self.contenu)

    # pour représenter la Pile
    def __repr__(self):
    ch = ""
    for e in self.contenu:
        ch = str(e) + "," + ch # ne pas oublier de convertir le
    ch = ch[:-1] # pour enlever la dernière virgule
    ch = ">" + ch+']'
    return ch
```

Pour simplifier, l’opération defiler renverra également le premier élément (en plus de le retirer de la file). L’opération premier n’est alors plus nécessaire. Vous devez donc implémenter une classe File permettant les opérations suivantes :

- création d’une file vide
- enfiler : ajout en queue de file
- defiler : renvoie le premier élement de la file et retire cet élément de la file
- len : accès au nombre d’éléments

Aide :

- Opération enfiler (simple) : C'est toujours dans l'une des deux piles (par exemple pA) que l'on empile un nouvel élément à enfiler.
- Opération defiler :
- Si l'autre pile (pB) n'est pas vide, son sommet est le premier élément de la file (celui à défiler)

- Sinon (si pB est vide), le premier élément de la file (celui à défiler) est au fond de pA. On peut alors "retourner" pA sur pB pour le premier élément de la file arrive au sommet de pB.

- Opération len (simple) : il suffit d'utiliser la méthode taille définie dans la classe Pile.

Compléter les pointillés ...

```
class File:
    """File avec deux piles"""
    def __init__(self):
        self.pA = Pile() # pA et pB sont les deux attributs de
        self.pB = Pile()

    def enfiler(self, e):
        return self.pA.empiler(e)

    def __len__(self):
        return ....

    def defiler(self):
        if self.pA.taille() == 0 and self.pB.taille() == 0:
            raise ValueError("on ne peut pas défiler une file v
        # La méthode __repr__ est définie pour que vous puissie
        else:
            if self.pB.taille() == 0:
                for x in range(....):
                    x=....
                    self.pB.empiler(x)
                    ....
                self.pA.depiler()
                print('pA',self.pA)
                print('pB1',self.pB)
            else:
                ....
```

```
    def __repr__(self):
        import copy
        #print("pile A : ", repr(self.pA)) # pour voir le conte
        #print("pile B : ", repr(self.pB))
        lstA = copy.copy(self.pA.contenu) # copie des list Pyth
        lstB = copy.copy(self.pB.contenu) # pour ne pas les mod
        lstB.reverse() # on a besoin de renverser lstB pour avo
        lst = lstB + lstA # et de concaténer lstB et lstA dans

        # on construit ensuite la chaine "<...<" qui représente
        ch = ""
        for e in lst:
            ch = ch + str(e) + ","
        ch = ch[:-1] # pour enlever la dernière virgule
        ch = "<" + ch + "<"
        return ch
```

Exercice n°3 :

Exercice n°2 :

Les variables `liste_eleves` et `liste_notes` ayant été préalablement définies et étant de même longueur, la fonction `meilleures_notes` renvoie la note maximale qui a été attribuée, le nombre d'élèves ayant obtenu cette note et la liste des noms de ces élèves.

Compléter le code Python de la fonction `meilleures_notes` ci-dessous.

```
liste_eleves = ['a','b','c','d','e','f','g','h','i','j']
liste_notes = [1, 40, 80, 60, 58, 80, 75, 80, 60, 24]

def meilleures_notes():
    note_maxi = 0
    nb_eleves_note_maxi = ...
    liste_maxi = ...

    for compteur in range(...):
        if liste_notes[compteur] == ...:
            nb_eleves_note_maxi = nb_eleves_note_maxi + 1
            liste_maxi.append(liste_eleves[...])
        if liste_notes[compteur] > note_maxi:
```

```
            note_maxi = liste_notes[compteur]
            nb_eleves_note_maxi = ...
            liste_maxi = [...]

    return (note_maxi,nb_eleves_note_maxi,liste_maxi)
```

Une fois complété, le code ci-dessus donne

```
>>> meilleures_notes()
(80, 3, ['c', 'f', 'h'])
```

```
liste_eleves = ['a','b','c','d','e','f','g','h','i','j'] liste_

def meilleures_notes():
    note_maxi = 0
    nb_eleves_note_maxi = ...
    liste_maxi = ...
    for compteur in range(...):
        if liste_notes[compteur] == ...:
            nb_eleves_note_maxi = nb_eleves_note_maxi + 1
            liste_maxi.append(liste_eleves[...])
        if liste_notes[compteur] > note_maxi:
            note_maxi = liste_notes[compteur]
            nb_eleves_note_maxi = ...
            liste_maxi = [...]

    return (note_maxi,nb_eleves_note_maxi,liste_maxi)
```