DM : POO, Test de programme, Récursivité

Nom :	**Note : / 20**
Ce travail est à rédiger numériquement en intégrant les copies de code et résultats python si nécessaire.	**Classe :**

1 Programmation orientée objet

Q1. Quelles sont les valeurs affichées par ce programme ?

a) 10, 20

b) 10, 10

c) 10, 100

d) 100, 10

e) 100, 100

```
def mystere(p):
    p['x'] = 100
    return p

p1 = {'x':100, 'y':20}
p2 = mystere(p1)

print(p1['x'], p2['x'])
```

Q2. Combien d'erreurs dans ce code ?

```
class Personne:
    def __init__(self , birthyear , tall , weight)
        self.birthyear = birthyear
        self.tall = tall
        self.weight = weight

    def grandir(h):
        self.tall += h


luc = personne(2002,178,69)
```

Q3. Expliquez en détail ce que permet d'afficher ce programme.

```
import random

class Piece :
    def alea(self) :
        return random.randint(0,1)

    def moyenne(self, n):
        tirage = [ ]
        for i in range (n) :
            tirage.append( self.alea() )

        return sum(tirage) / n

p = Piece()
print( p.moyenne(100) )
```

2 Mise au point de programme

Tests unitaires

Un test unitaire consiste à vérifier qu'une unité de programme, par exemple une fonction, se comporte comme prévu. On distingue les tests 'boite noire', définis à partir de la seule spécification du programme et les tests 'boite blanche' conçus à partir du code.

Les tests seront réalisés avec des **assert**.

Q4. Partage de tableau

On souhaite définir la fonction **partage(t)** qui partage le tableau **t** en deux tableaux contenant respectivement les éléments de **t** de rang pair et de rang impair et retourne ces deux tableaux sous forme de tuple.

Exemple :

```
In [15]: t
Out[15]: ['a', 'b', 3, '6']

In [16]: partage(t)
Out[16]: (['a', 3], ['b', '6'])
```

```
def partage(t):
    assert isinstance(t, list)
    pairs = []
    impairs = []
    i = 0
    while i != len(t):
        pairs.append(t[i])
        impairs.append(t[i+1])
        i += 2
    return pairs, impairs
```

1) A partir de cette spécification écrire un jeu de tests 'boite noire' de cette fonction.

2) Soit une proposition de solution pour la fonction : est-ce qu'elle passe tous vos tests ?

3) Écrire un test 'boite blanche' qui échoue.

4) Corriger la fonction pour qu'elle passe tous les tests.

Q5. Fonction mystère

1) Que fait cette fonction ?

```
def mystere2(x):
    y = 0
    for z in x:
        if z > y:
            y = z
    return y
```

2) Réécrire cette fonction en donnant des noms expressifs aux variables utilisées.

3) Écrire au moins un test qui produit un résultat erroné.

4) Corriger cette fonction pour qu'elle passe ce test.

3 Programmation récursive

Q6. Multiplication sans addition

On suppose que l'opération de multiplication n'ait pas été définie en python. On souhaite créer une fonction récursive **mult(x, y)** qui calcule la multiplication de deux nombres entiers en utilisant uniquement des additions avec l'identité mathématique ci-dessous :

$$x \cdot y = x \cdot (y-1) + x \quad \text{pour } y > 0$$

1) Identifier la récursivité en précisant le cas de base et le cas récursif, faire un schéma explicatif comme sur le polycopié de cours.

2) Écrire le code python de cette fonction.

3) Réaliser les tests 'boites noires' de votre fonction.

4) Combien d'additions de nombres effectue l'algorithme pour multiplier x et y ?