NOM : _____ Prénom : _____

Interrogation écrite
Tester et déboguer

Durée : 30 minutes

Les calculatrices ne sont pas autorisées.

Consignes. Le sujet est composé de deux exercice qu'il faut traiter en intégralité. Certaines questions sont à réaliser en complétant directement l'énoncé, d'autre questions doivent être rédigées intégralement sur la copie. Il est rappelé que la qualité de la rédaction, la clarté et la précision des raisonnements entreront pour une part importante dans l'appréciation des copies. Rendre le sujet avec la copie.

Exercice 1. Corriger un bogue (6 points)

1. On donne ci-dessous la définition d'une fonction `mystere` qui prend en paramètre un nombre.

```
def mystere(x):
    assert isinstance(x, int)
    return 2 + x
```

On suppose que la définition de la fonction `mystere` est chargée en mémoire.

(a) Que se passe t'il si l'on exécute l'instruction `mystere(3)` ?

(b) Que se passe t'il si l'on exécute l'instruction `mystere(3.0)` ?

(c) Corriger l'assertion de la fonction `mystere` afin que tous les nombres (et seulement les nombres) soit acceptés comme argument lors d'un appel de cette fonction.

2. On donne ci-dessous la définition d'une fonction `mystere` prenant en paramètre un tableau Python (type `list`) dont tous les éléments sont des nombres.

```
def mystere(t):
    var = 0
    for i in range(1, len(t) + 1):
        var += t[i]
    return var
```

(a) Que tente de faire cette fonction ? *Aide.* Tester la fonction avec un exemple de tableau.

(b) Pourquoi cette fonction est-elle incorrecte ?

(c) Proposer une définition correcte de la fonction `mystere`. Plusieurs solutions sont possibles.

3. On considère le programme Python ci-dessous :

```
x = 0.1
while x != 1.0:
    print(x)
    x += 0.1
```

On souhaite que l'exécution de ce programme produise exactement 10 affichages successifs.

(a) Pourquoi ce programme est-il incorrect ?

(b) Proposer une version correcte de ce programme en modifiant uniquement sa seconde ligne.

Exercice 2. Spécification et tests (4 points)

On souhaite écrire une fonction Python `partage(t)` qui partage le tableau Python (type `list`) `t` en deux tableaux contenant respectivement les éléments de `t` de rang pair et impair et renvoie ces deux tableaux sous forme d'un tuple. On donne ci-dessous un exemple de résultat attendu dans la console Python :

```
>>> partage([7, 5, 2, 1, 8, 3])
([7, 2, 8], [5, 1, 3])
```

Remarques.

— `t` peut contenir un nombre impair d'élément(s).

— `t` peut être un tableau vide.

1. Avant d'écrire la fonction `partage`, on a préparé un module de test à l'aide du module `pytest`. On donne ci-dessous le code de ce module de test incomplet.

```
import pytest

def test_precondition():
    with pytest.raises(AssertionError):                    # test n°1
        partage(1)
    with pytest.raises(AssertionError):                    # test n°2
        partage('mot')

def jeu_de_test():
    assert partage([7, 5, 2, 1, 8, 3]) == ...              # test n°3
    assert partage([7, 5, 2, 1, 8]) == ([7, 2, 8], [5, 1])  # test n°4
    assert partage([]) == ([], [])                         # test n°5
```

   Compléter ci-dessus le test n°3 dans la fonction `jeu_de_test` en remplaçant les `...` par le code pertinent.

2. Un programmeur a écrit l'implémentation suivante de la fonction `partage`.

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

   Dans les questions suivantes, on ne tient pas compte du test n°3 du module de test.

   (a) Avec l'implémentation précédente, indiquer le numéro de chaque test que la fonction `partage` ne passe pas.

   (b) Proposer une implémentation modifiée de la fonction `partage` afin qu'elle passe tous les tests.