Exercices spécifications et tests

Exercice 1 : Prototypage

Prototyper correctement les fonctions suivantes. Il est possible de les tester pour mieux les reconnaître.

```
# Mystère 1
def a(b, c, d, e):
    f = (b + c + d + e) / 4
    return f
```

```
# Mystère 2
def a(b):
    if b%2 == 0:
        return True
    else:
        return False
```

```
# Mystère 3
def a(b, c):
    if b >= c:
        return b
    else:
        return c
```

```
# Mystère 4
def a(b, c):
    d = 0
    for _ in range(c):
        d = d + b
    return d
```

```
# Mystère 5
def a(b, c):
    d = b // c
    e = b % c
    return (d, e)
```

```
# Mystère 6
def a(b, c, d):
    b = b**2
    c = c**2
    d = d**2

    e = b == c + d
    e = e or c == b + d
    e = e or d == b + c
    return e
```

```
# Mystère 7
def a(b, c, d):
    e = b[c]
    b[c] = b[d]
    b[d] = e
```

```
# Mystère 8
def a(b):
    c = []
    while b > 0:
        if b%2 == 0:
            c.insert(0, 0)
            b = b // 2
        else:
            c.insert(0, 1)
            b = (b - 1) // 2
    return c
```

Exercice 2 : Typage.

Typer du mieux possible les fonctions de l'exercice 1.

Exercice 3 : Chaîne de documentation.

Créer les chaînes de documentation des fonctions de l'exercice 1. Elle doivent être cohérentes avec vos typages.

Exercice 4 : Tests de post-conditions.

Écrire deux tests pertinents (post-conditions) avec des assertions pour chacune des fonctions de l'exercice 1.
Ils doivent être cohérent avec vos typages.

Exercice 5 : Tests de préconditions.

Pour chacune des fonctions de l'exercice 1, écrire 2 tests de préconditions sur les arguments, cohérents avec vos typages.

Exercice 6 : Doctest.

Intégrer vos tests de l'exercice 4 à la chaîne de documentation des fonctions et utiliser le module doctest.

Exercice 7 : Type checker

Écrire pour chacune des fonctions avec typage, un appel où le typage est respecté et un autre où il ne l'est pas puis utiliser mypy sur le site suivant pour analyser votre code. https://mypy-play.net/