Informatique PCSI
Corrigé TP 10a : analyse d'un programme

Exercice 1

En sortie de boucle la valeur de `k` est 0. C'est donc la proposition 2 qui est fausse.

Exercice 2

On définit deux fonctions et on utilise l'affichage avec `print`.

```
def g(x):
    return x

print(dichotomie(g, -1, 1, 0.001))

def h(x):
    return x**3 - 5*x**2 + x - 5

print(dichotomie(h, 0, 10, 0.001))
```

Dans les deux cas, on obtient pour solution une valeur approchée de `b` à la place de 0 pour la première équation et à la place de 5 pour la deuxième équation.

On peut ajouter une assertion dans la boucle, `assert f(a) * f(b) <= 0`, pour constater qu'une propriété qui doit être invariante ne l'est pas. En effet si `f(a)` vaut 0, alors `a` prend la valeur de `m` non nulle et du même signe que la valeur de `b`. Ce cas se produit aussi si `f(m)` vaut 0 puisque alors, `a` prend la valeur de `m` et au passage suivant dans la boucle `f(a)` a la valeur 0.

Il faut remplacer `if f(a) * f(m) < 0` par `if f(a) * f(m) <= 0`.

Exercice 3

1. La fonction `test1` prend en paramètre une fonction `f`.

```
    def test1(f):
        liste = [2, 3, 5, 7, 11, 13, 17, 19]
        for n in liste:
            if not f(n):
                return False
        return True
```

2. La fonction `premier` passe ces tests sans problème : `test1(premier)` renvoie `True`.

3. On définit la fonction `test2` en suivant le modèle de la fonction `test1`.
   On constate un problème : `test2(premier)` renvoie `False`.

```
    def test2(f):
        liste = [0, 1, 4, 6, 8, 9, 10]
        for n in liste:
            if f(n):
```

```
                return False
        return True
```

4. À l'examen du code nous pouvons remarquer que la dernière ligne `return True` de la fonction `premier` est mal indentée. Seul le premier passage est exécuté dans la boucle `for`.

Exercice 4

La terminaison est évidente avec cette boucle for.

La correction se prouve en considérant la propriété pour `i` prenant les valeurs de 1 à `len(tab)-1` : « `maxi` est le plus grand élément de la liste `tab[0: i+1]` ».

Après le premier passage dans la boucle, avec `i` égal à 1, `maxi` est le plus grand élément de la liste `tab[0:2]` qui contient `tab[0]` et `tab[1]`.

Si la propriété est vraie après un passage, `maxi` est le plus grand élément de `tab[0:i+1]`, et au passage suivant l'élément `tab[i+1]` est comparé à `maxi` qui prend alors la valeur du plus grand des deux. Donc `maxi` est le plus grand élément de la liste `tab[0:i+2]`.

En sortie de boucle, `maxi` est le plus grand élément de `tab[0:len(tab)]`, donc de `tab`.

Exercice 5

1. La suite formée par les valeurs de r au cours des itérations est une suite d'entiers strictement décroissante : r étant initialisée à m, si $m \geq n$ alors la valeur de r est inférieure à celle de n en un maximum de $m - n$ étapes et la boucle est terminée.
2. Les valeurs initiales q et r sont respectivement à 0 et m. Donc la propriété $m = nq + r = n.0 + m$ est vérifiée avant le premier passage dans la boucle.

   Avant une itération arbitraire, supposons que l'on ait $m = nq + r$ et montrons que cette propriété est encore vraie après cette itération. Soient q' la valeur de q et r' la valeur de r à la fin de l'itération. Nous devons montrer que $m = nq' + r'$.

   On a $q' = q + 1$ et $r' = r - n$, alors $nq' + r' = n(q + 1) + (r - n) = nq + r = m$. La propriété est bien conservée.

   En sortie de boucle, la propriété $m = nq+r$ est vérifiée, mais ce n'est pas suffisant pour une division euclidienne. Le reste r doit vérifier la condition $0 \leq r < n$.

   Or, si le programme s'arrête, c'est que la condition $r > n$ n'est plus satisfaite, donc que $r \leq n$ et l'égalité $r = n$ reste possible. On obtient ce cas si m est un multiple de n. On corrige donc le code en remplaçant `r > n` par `r >= n`.

   Il reste à montrer qu'en fin de boucle $r \geq 0$. Comme r est diminuée de n à chaque itération, à l'itération précédente la valeur de r est $r' = r + n$. Donc si $r < 0$ alors $r' < n$. Et donc la boucle se serait arrêtée à l'itération précédente, ce qui est absurde. On en déduit donc que $r \geq 0$.

   En conclusion, le programme se termine avec $0 \leq r < n$ et la propriété $m = nq + r$ est vérifiée à chaque itération ; ceci prouve que l'algorithme effectue bien la division euclidienne de m par n.
3. On envisage pour m les cas $m = 0$, $m = 1$ et $m > 1$. Pour n, les cas sont $n = 1$ ou $n > 1$. Ensuite, pour les couples $(m; n)$, on peut envisager $m < n$, $m = n$, $m > n$, m multiple de n ou m diviseur de n. Ces test auraient permis de détecter l'erreur dans la fonction `div_euclid`.

```
assert div_euclid2(0, 3) == (0, 0)
assert div_euclid2(1, 1) == (1, 0)
assert div_euclid2(1, 3) == (0, 1)
assert div_euclid2(4, 1) == (4, 0)
assert div_euclid2(3, 8) == (0, 3)
```

```
    assert div_euclid2(3, 3) == (1, 0)
    assert div_euclid2(8, 3) == (2, 2)
    assert div_euclid2(6, 2) == (3, 0)
    assert div_euclid2(2, 6) == (0, 2)
```