Exercice 1 :

```
#Question 1
def applique(f, t: tuple)->tuple:
    return tuple(f(x) for x in t)

#Question 2
def double(x: int)->int:
    return x*2

#Question 3
l = (2,4,5,8,10)
print(applique(double,l))
```

Exercice 2 :

```
from time import sleep
import turtle as t

#Question 1
def repeter_delai(f, n: int, t: int)->None:
    for i in range(n):
        f(i)
        sleep(t)

#Question 2
def dessine(n: int)->None:
    t.forward(n*50)
    t.left(90)

#Question 3
repeter_delai(dessine, 10, 1)
t.bye()
```

Exercice 3 :

```
#Question 1
def trouve(p,t: tuple)->object:
    for x in t:
        if p(x):
            return x
    return None

#Question 2
def est_positif(x: int)->bool:
    if x > 0:
        return True
    return False

#Question 3
l = (-6,-5,12,4,8)
print(trouve(est_positif, l))

```


Christophe Viroulaud

```python
#Question 4
from math import sqrt

def est_premier(n: int)->bool:
    if n ==1:
        return False
    if n == 2:
        return True
    if n%2 == 0: #nombre pair; solution avec bitwise operator: n & 1 == 0
        return False

    k = 3
    racine = int(sqrt(n)) #renvoie racine entière (int)
    while k <= racine:
        if n % k == 0:
            return False
        k += 2
    return True

#Question 5
l1 = (4, 8, 22, 35, 47, 87)
print(trouve(est_premier, l1))
```

Exercice 4 :

1. Un invariant de boucle est une propriété qui est vraie avant chaque itération de la boucle et qui reste vraie après. Il permet de prouver la correction de l'algorithme.
2. Avant la première itération, i vaut 1. Le tableau [O:1] ne contient qu'une valeur, il est donc trié.
3. Raisonnement par récurrence :
   - **initialisation :** La propriété est vraie avant la première itération (question précédente.
   - **hérédité :** Considérons la propriété vraie au rang k. Le tableau [0:k] est trié.
   - **conclusion :** À l'itération *n-1* (dernière itération) la boucle *while* insère à la bonne place le *n-ème* élément dans le tableau [0:n-1] déjà trié. Le tableau [0:n] est donc trié.

Exercice 5 :

```python
def h(f,g):
    return lambda x: f(g(x))

f = lambda x: x**2
g = lambda x: 2*x+3

print(h(f,g)(5))
```

Exercice 6 :

```python
#Question 1
def calcul(operation, l: tuple)->int:
    res = l[0]
    for i in range(1, len(l)):
        res = operation(res, l[i])
```


Christophe Viroulaud

```
6       return res
7
8   #Question 2
9   addition = lambda x,y: x+y
10
11  #Question 3
12  l = (2,3,6,8,1,9)
13  print(calcul(addition, l))
14
15  #Question 4
16  print(calcul(lambda x,y: x-y, l))
17  print(calcul(lambda x,y: max(x,y), l))
```


Christophe Viroulaud