

**Exercice 1 :**

```
1 #Question 1
2 def applique(f, t: tuple)->tuple:
3     return tuple(f(x) for x in t)
4
5 #Question 2
6 def double(x: int)->int:
7     return x*x
8
9 #Question 3
10 l = (2,4,5,8,10)
11 print(applique(double,l))
```

**Exercice 2 :**

```
1 from time import sleep
2 import turtle as t
3
4 #Question 1
5 def repeter_delai(f, n: int, t: int)->None:
6     for i in range(n):
7         f(i)
8         sleep(t)
9
10 #Question 2
11 def dessine(n: int)->None:
12     t.forward(n*50)
13     t.left(90)
14
15 #Question 3
16 repeter_delai(dessine, 10, 1)
17 t.bye()
```

**Exercice 3 :**

```
1 #Question 1
2 def trouve(p,t: tuple)->object:
3     for x in t:
4         if p(x):
5             return x
6     return None
7
8 #Question 2
9 def est_positif(x: int)->bool:
10     if x > 0:
11         return True
12     return False
13
14 #Question 3
15 l = (-6,-5,12,4,8)
16 print(trouve(est_positif, l))
17
```

```

18 #Question 4
19 from math import sqrt
20
21 def est_premier(n: int)->bool:
22     if n ==1:
23         return False
24     if n == 2:
25         return True
26     if n%2 == 0: #nombre pair; solution avec bitwise operator: n
27         & 1 == 0
28         return False
29
30     k = 3
31     racine = int(sqrt(n)) #renvoie racine entière (int)
32     while k <= racine:
33         if n % k == 0:
34             return False
35         k += 2
36     return True
37
38 #Question 5
39 l1 = (4, 8, 22, 35, 47, 87)
40 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  $[0:1]$  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 :**

```

1 def h(f,g):
2     return lambda x: f(g(x))
3
4 f = lambda x: x**2
5 g = lambda x: 2*x+3
6
7 print(h(f,g)(5))

```

**Exercice 6 :**

```

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

```

```
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))
```