On se propose de tester l'algorithme qui permet de faire la division euclidienne de a par b en soustrayant itérativement b à a et en comptant le nombre de soustractions nécessaires pour que le reste soit compris entre 0 et b(exclu).

1. Compléter le code ci-dessous .

```
def division_euclidienne(a , b):
    '''(int,int) -> tuple(int,int)
    Preconditions: a >= 0 et b > 0 avec a et b entiers.
    Si les préconditions ne sont pas respectées, doit renvoyer -1.
    Fonction effectuant la division euclidienne de a par b en
   utilisant
    l'algorithme par soustraction. Renvoie un doublet (quotient,
   reste), de
    sorte que a = quotient*b + reste avec 0 <= reste < b.
    '''

    #on écrit ici le code
    #.....
    return None  #remplacer none par ce qui convient
```

2. La fonction est supposée renvoyer un tuple contenant respectivement le quotient q et le reste r de la division euclidienne de a par b. S'il n'est pas possible d'obtenir le résultat demandé par l'algorithme proposé (par exemple si a ou b sont négatifs) ou que les arguments donnés à la fonction ne sont pas entiers, on choisit de renvoyer -1.

   On peut utiliser assert pour vérifier l'exécution du programme dans des cas "anormaux" comme ci-dessous.

```
def division_euclidienne(a , b):
    '''(int,int) -> tuple(int,int)
    Preconditions: a >= 0 et b > 0 avec a et b entiers.
    Si les préconditions ne sont pas respectées, doit renvoyer -1.
    Fonction effectuant la division euclidienne de a par b en
   utilisant
    l'algorithme par soustraction. Renvoie un doublet (quotient,
   reste), de
    sorte que a = quotient*b + reste avec 0 <= reste < b.
    '''

    #on écrit ici le code
    #.....
    return None  #remplacer none par ce qui convient


    if __name__ == '__main__':
        assert division_euclidienne(10, 2) == (5, 0)
        assert division_euclidienne(2, 10) == (0, 2)
        assert division_euclidienne(37, 3) == (12, 1)
        assert division_euclidienne(-10, 7) == -1
        assert division_euclidienne(10, -7) == -1
```

```
assert division_euclidienne(10.3, 4) == -1
assert division_euclidienne(11, 3.5) == -1
assert division_euclidienne(3, 0) == -1
assert division_euclidienne(0, 3) == (0, 0)
assert division_euclidienne(0, 0) == -1
```

'inconvénient majeur de cette méthode (même si cela peut en fait être un avantage) est que l'on doit traiter une assertion après l'autre car le programme s'arrête dès le premier test qui échoue. En cas de réussite, il ne se passe simplement rien.

3. Utiliser le module doctest pour réaliser les tests précédents dans la spécification.