1 Sans faire de preuve précise, expliquer si les boucles suivantes se terminent.

1.
```
for i in range(10 ** 30):
    print(i)
```

2.
```
i = 0
while i < 10:
    i = i - 1
```

3.
```
i = 0
while True:
    i = i + 1
    print(i)
```

4.
```
n = 100
i = 0
while i < n:
    i = i + 2
    n = n + 1
```

5.
```
n = 10
i = 0
while i < n:
    i = i + 1
    n = n + 1
```

6.
```
i = 1
while (i < 10) or (i % 2 == 1):
    i = i +2
```

7.
```
x = 1
while x / 2 >0:
    x = x / 2
```

8.
```
import random as rd
n = rd.randint(10 ** 300, 10 ** 400)
while n > 1:
    if n % 2 == 0:
        n = n // 2
    else:
        n = 3 * n + 1
```

2 On considère la fonction suivante :

```
def fonction(n):
    # n est un entier naturel
    r = 2
    i = 0
    while i < n:
        r = r * r
        i = i + 1
    return(r)
```

1. En faisant des essais, dire ce que calcule la fonction.
2. À l'aide d'un variant de boucle, justifier que la fonction se termine.
3. Pour $i \in [\![0, n]\!]$, on note r_i la valeur de r à la fin de la i-ème itération de la boucle while, avec par convention $r_0 = 2$. Démontrer que la propriété suivante est un invariant de boucle :

$$P_i \ : \ r_i = 2^{2^i}$$

Justifier alors la correction de la fonction.

3 Soit p un entier naturel choisi par l'utilisateur, on considère le programme suivant :

```
c = 0
while p > 0:
    if c == 0:
        p = p - 2
        c = 1
    else:
        p = p + 1
        c = 0
```

Justifier que ce programme se termine en démontrant que $2p + 3c$ est un variant de boucle.

4 On considère la fonction suivante qui calcule la factorielle d'un entier naturel.

```
def fact(n):
    """renvoie n!"""
    p = 1
    for i in range(1, n + 1):
        p = p * i
    return(p)
```

1. Démontrer que la fonction se termine.
2. En trouvant un invariant de boucle, justifier la correction de cet algorithme.

5 **Exponentiation rapide.** Soit $n \in \mathbb{N}^*$ et a un nombre flottant donnés par l'utilisateur, on considère le programme suivant :

```
N = n
A = a
R = 1
while N > 0:
    if N % 2 == 0:
        A = A * A
        N = N // 2
    else:
        R = R * A
        N = N - 1
print(R)
```

1. En suivant pas à pas l'algorithme pour différentes valeurs de n, expliquer ce que renvoie ce programme.
2. Démontrer que l'algorithme se termine.
3. Justifier que la propriété P : $A^N \times R = a^n$ est un invariant de boucle et conclure quant à la correction de l'algorithme.

6 Voici un test utilisé en calcul mental afin de savoir si un nombre entier naturel est divisible par 7. Soit $n \in \mathbb{N}$, on enlève à l'écriture décimale de n le chiffre des unités et on le retranche deux fois au nombre ainsi obtenu. On réitère le procédé jusqu'à ce que l'on puisse savoir de tête si le nombre obtenu est divisible par 7. Par exemple :

$$31976 \to 3197 - 2 \times 6 = 3185 \to 318 - 2 \times 5 = 308 \to 30 - 2 \times 8 = 14$$

Ici 14 est divisible par 7, on en déduit que 31976 également. La réciproque est vraie, c'est-à-dire que si l'on tombe sur un nombre qui n'est pas divisible par 7 alors on peut en conclure que le nombre de départ ne l'est pas non plus.

1. Écrire un algorithme qui effectue ce procédé et s'arrête lorsque le nombre de départ est inférieur ou égal à 70 (on considère alors que le caractère divisible ou non par 7 est clair).
2. Démontrer que votre algorithme se termine.
3. Trouver un invariant de boucle.