TD : TERMINAISON ET CORRECTION

1. Étudiez la terminaison de chacune des boucles suivantes. Si elles ne terminent pas, exhiber une situation particulière suffit. Dans le cas contraire, une démonstration formelle de terminaison avec un variant de boucle est attendue.

```
-- 1 : n ∈ ℤ⁻
POUR i ALLANT DE n À -1 FAIRE
    afficher (i)
FIN POUR
```

```
-- 2 : n ∈ ℕ, m ∈ ℕ
TANT QUE n ≠ 0 FAIRE
    n ← n - m
FIN TANT QUE
```

```
-- 3 : n ∈ ℕ, T tableau d'entiers
TANT QUE n ≠ TAILLE(T) FAIRE
    n ← n + 1
FIN TANT QUE
```

```
-- 4 :  a ∈ ℕ, b ∈ ℕ
TANT QUE a > 0 ET b > 0 FAIRE
    SI a ≡ 0 (mod 2) ALORS
        a ← a + 1
        b ← b - 3
    SINON
        a ← a - 2
        b ← b + 1
    FIN SI
FIN TANT QUE
```

```
-- 5 : a ∈ ℤ, b ∈ ℤ, c ∈ ℤ
TANT QUE a > 0 ET b > 0 FAIRE
    SI c ≡ 0 (mod 2) ALORS
        a ← a - c
        c ← c + 1
    SINON
        b ← b - c
        c ← c - 1
    FIN SI
FIN TANT QUE
```

```
-- 6 : T tableau d'entiers non vide, e ∈ ℕ
i ← e × TAILLE(T)
TANT QUE T_{i mod TAILLE(T)} ≠ e FAIRE
    i ← i + 1
FIN TANT QUE
```

```
--  7 : x ∈ ℕ, y ∈ ℕ
TANT QUE x ≠ 0 OU y ≠ 0 FAIRE
    SI y > 0 ALORS
        y ← y - 1
    SINON
        y ← entier aléatoire compris entre 0 et x-1
        x ← 0
    FIN SI
FIN TANT QUE
```

2. Montrez la terminaison de l'algorithme suivant à l'aide du variant de boucle `TAILLE(T)-i`, puis montrez sa correction à l'aide de l'invariant `Inv(i, res): res` $= \sum_{\substack{0 \leqslant j < i \\ T_j = e}} 1$.

```
NB_OCCS(T tableau d'entiers, e ∈ ℤ)
    res ← 0
    POUR i ALLANT DE 0 À TAILLE(T) - 1 FAIRE
        SI T_i = e ALORS
            res ← res + 1
        FIN SI
    FIN POUR
    RENVOYER res
```

3. Que fait l'algorithme suivant ? En posant $N \leftarrow$ `n` avant la boucle, montrez la correction partielle de l'algorithme avec l'invariant `Inv(n): n` $\equiv N \pmod 2$. La correction est-elle totale (montrez-le) ?

```
MYSTERE(n ∈ ℕ)
    TANT QUE n ≠ 0 ET n ≠ 1 FAIRE
        n ← n - 2
    FIN TANT QUE
    RENVOYER n = 0
```

4. L'algorithme suivant détermine le plus petit entier k tel que $n \leqslant 2^k$. Montrez que l'algorithme est partiellement correct avec l'invariant `Inv(k, p)` : ($p = 2^k$ et $2^{k-1} < n$). La correction est-elle totale (montrez-le) ?

```
LOG(n ∈ ℕ*)
    k ← 0
    p ← 1
    TANT QUE p < n FAIRE
        k ← k + 1
        p ← 2 × p
    FIN TANT QUE
    RENVOYER k
```

5. Montrez la correction totale de l'algorithme suivant :

```
MAX_TAB(T tableau non vide d'entiers)
    maxi ← T_0
    POUR i ALLANT DE 1 À TAILLE(T)-1 FAIRE
        SI maxi < T_i ALORS
            maxi = T_i
        FIN SI
    FIN POUR
    RENVOYER maxi
```

6. Montrez que l'algorithme suivant termine, puis en exhibant un invariant, montrez qu'il renvoie le `n`-ième terme de la suite de Fibonacci.

```
FIBO(n ∈ ℕ)
    avant_dernier ← 0
    dernier ← 1
    POUR i allant de 0 à n-1 FAIRE
        tmp ← avant_dernier + dernier
        avant_dernier ← dernier
        dernier ← tmp
    FIN POUR
    RENVOYER avant_dernier
```

7. Montrez que l'algorithme suivant termine, puis en exhibant un invariant, montrez qu'il détermine si deux tableaux triés dans l'ordre croissant sont disjoints.

```
DISJOINTS(U, V tableaux triés d'entiers)
    iu ← 0
    iv ← 0
    TANT QUE iu < TAILLE(U) ET iv < TAILLE(V) ET U_iu ≠ V_iv FAIRE
        SI U_iu < V_iv ALORS
            iu ← iu + 1
        SINON
            iv ← iv + 1
        FIN SI
    FIN TANT QUE
    RENVOYER iu = TAILLE(U) OU iv = TAILLE(V)
```

8. Étudiez la terminaison des fonctions récursives suivantes. Si elles ne terminent pas, exhiber une situation particulière suffit. Dans le cas contraire, une démonstration formelle de terminaison est attendue.

```
-- 1 : F (n ∈ ℕ, m ∈ ℕ)
SI n ≠ 0 ALORS
    F (n-m, m)
FIN SI
```

```
-- 2 : F (x ∈ ℕ, y ∈ ℕ)
SI x = 0 ET y = 0 ALORS
    RENVOYER 0
SINON SI x > 0 ALORS
    RENVOYER 1 + F(x-1, y)
SINON
    RENVOYER 1 + F(x, y-1)
FIN SI
```

```
-- 3 : F (x ∈ ℕ, y ∈ ℕ)
SI x ≤ 0 ET y ≤ 0 ALORS
    RENVOYER 2
SINON SI x > 0 ALORS
    RENVOYER 3 + F(x-y-3, 2y)
SINON
    RENVOYER -3 + F(x+5, y-8)
FIN SI
```

```
--  4 : F (L liste d'entiers)
SI L est vide ALORS
    RENVOYER VRAI
SINON SI queue(L) est vide ALORS
    RENVOYER FAUX
SINON SI tete(L) ≠ tete(queue(L)) ALORS
    RENVOYER FAUX
SINON
    RENVOYER F (queue(queue(L)))
FIN SI
```

```
-- 5 : F (a ∈ ℤ, b ∈ ℤ)
SI a = b ALORS
    RENVOYER VRAI
SI a < 0 OU b < 0 ALORS
    RENVOYER FAUX
SI a est pair ALORS
    RENVOYER F(b, a)
SINON
    RENVOYER F(a-1, b-1)
FIN SI
```

```
--  6 : F (a ∈ ℤ, b ∈ ℤ)
SI a = b ALORS
    RENVOYER VRAI
SI a < 0 ET b < 0 ALORS
    RENVOYER FAUX
SI a est pair ALORS
    RENVOYER F(a-1, b+1)
SINON
    RENVOYER F(a-1, b-1)
FIN SI
```

```
-- 7 : F (x ∈ ℤ, y ∈ ℤ)
SI x = 0 ET y = 0 ALORS
    RENVOYER VRAI
SINON SI x > 0 ALORS
    RENVOYER F(x-1, y)
SINON SI y > 0 ALORS
    RENVOYER F(x, y-1)
SINON
    tmp ← entier aléatoire quelconque
    RENVOYER F(x+tmp, y+tmp)
FIN SI
```

9. Montrez la correction totale de chacune des fonctions récursives suivantes.

```
-- 1 : x_puissance_n (x ∈ ℕ, n ∈ ℕ)
SI n = 0 ALORS
    RENVOYER 1
SINON
    RENVOYER x * x_puissance_n (x, n-1)
FIN SI
```

```
-- 2 : est_pair (n ∈ ℕ)
SI n = 0 ALORS
    RENVOYER VRAI
SINON SI n = 1 ALORS
    RENVOYER FAUX
SINON
    RENVOYER est_pair (n - 2)
FIN SI
```

```
-- 3 : somme_liste (L liste d'entiers)
SI L est vide ALORS
    RENVOYER 0
SINON
    RENVOYER tête(L) + somme_liste(queue(L))
FIN SI


-- 4 : minimum_liste (L liste non vide d'entiers)
SI queue(L) est vide ALORS
    RENVOYER tête(L)
SINON
    RENVOYER MIN (tête(L),  minimum_liste(queue(L)))
FIN SI

-- 5 : taille_l1_inferieure_taille_l2 (l1, l2 listes d'entiers)
SI l1 est vide ALORS
    RENVOYER VRAI
SI l2 est vide ALORS
    RENVOYER FAUX
SINON
    RENVOYER taille_l1_inferieure_taille_l2(queue(l1), queue(l2))
FIN SI

--  6 : disjoints (l1, l2 listes d'entiers)
SI l1 est vide OU l2 est vide ALORS
    RENVOYER VRAI
SINON
    RENVOYER (  tete(l1) ≠ tete(l2)
                ET F(queue(l1), l2)
                ET F(l1, queue(l2))  )
FIN SI
```

10. Pour chacun des algorithmes que vous devez maîtriser à ce stade de l'année, écrivez-le en pseudo-code (une version impérative et une fonctionnelle) et montrez sa correction totale.
    En voici quelques uns :
    - Calculer un PGCD avec l'algorithme d'Euclide.
    - Calculer le nombre de chiffres d'un entier naturel.
    - Trouver le nombre d'occurrences d'un élément dans une liste (fonctionnel) / un tableau (impératif).
    - Déterminer si une liste / un tableau est rangé(e) dans l'ordre croissant.
    - Trouver l'indice du maximum ou minimum d'une liste / d'un tableau.
    - …
11. Si vous avez terminé, implémentez les algorithmes de la question précédente en OCaml (fonctionnel) / en C (impératif).