Problème de la semaine 9

Exercice 1 — (Arithmétique).

Déterminer l'ensemble des entiers relatifs x tels que : $\begin{cases} x \equiv 3 \quad [10] \\ x \equiv 7 \quad [12] \end{cases}$

Exercice 2 — (Arithmétique, Python et complexité).

Partie 1 : test de primalité et complexité

1/ Le code ci-dessous est celui d'une fonction qui reçoit comme paramètre un entier $n \geqslant 2$, et qui doit tester ce nombre est premier. Explicitement, cette fonction doit renvoyer True lorsque l'entier n est premier, et False sinon.

Compléter le code pour qu'il réponde à la question.

```
def TpremOuPas(n):
    assert n >1
    TPREM =True
    d =2
    while (d <n) and (TPREM ==True):
        if  # LIGNE A COMPLETER
            TPREM =False
        d =  # LIGNE A COMPLETER
    return TPREM
```

2/ Une fois la fonction précédente complétée, que renvoie l'instruction :

```
[Tprem_ou_pas(k) for k in range(2,6)]
```

Et que renvoie l'instruction :

```
[Tprem_ou_pas(k) for k in range(1,6)]
```

3/ Quelle est la complexité algorithmique de la fonction `TpremOuPas` ? Linéaire, quadratique, exponentielle ?

4/ En partant du principe que votre ordinateur effectue 10^9 opérations par seconde, combien de temps peut prendre (au pire) la fonction TpremOuPas pour déterminer si le 59-ème nombre de Mersenne $(2^{59} - 1)$ est premier ou non ?[1]

5/ En modifiant une seule ligne du code de la fonction TpremOuPas, montrer que l'on peut rendre sa complexité racinaire.[2] Cette modification faite, combien de temps peut prendre (au pire) alors la fonction TpremOuPas pour déterminer si le 59-ème nombre de Mersenne est premier ou non ?

---

1. On pourra prendre comme ordre de grandeur : une journée $\approx 10^5$ secondes.
2. C'est-à-dire que sa complexité est un $O\left(\sqrt{n}\right)$

Partie 2 : valuation et complexité

On rappelle que, pour un nombre premier p et un entier $n \geqslant 2$ donnés, la valuation p-adique de n (notée $v_p(n)$) est la plus grande puissance de p divisant n.

6/ Ecrire une fonction `V3adic(n)` en Python qui reçoit comme paramètre un entier $n \geqslant 1$, et qui renvoie la valuation 3-adique de n.

7/ Montrer que la complexité algorithmique de cette fonction est logarithmique.

8/ Ecrire une fonction `Vadic(n,p)` en Python qui reçoit comme paramètre un entier $n \geqslant 1$, et un nombre premier p, et qui renvoie la valuation p-adique de n ; cette fonction doit également renvoyer un message d'erreur si $n < 1$, ou si p n'est pas premier.

9/ Quelle est la complexité algorithmique de cette fonction ?

Partie 3 : nombres de Carmichael

On rappelle qu'un entier de Carmichael est un entier $C \geqslant 4$, tel que :

1/ C n'est pas premier ; 2/ $\forall n \in [\![0, C-1]\!],\ n^C \equiv n \quad [C]$

Pour information, le plus petit entier de Carmichael est 561.

10/ Ecrire une fonction `Carmi(n)` en Python qui reçoit comme paramètre un entier $n \geqslant 4$, et qui renvoie True si n est un entier de Carmichael, et False sinon.

11/ Quelle est la complexité algorithmique de cette fonction ?

12/ Ecrire une fonction `ListCarmi(a,b)` en Python qui reçoit comme paramètre deux entiers $b > a \geqslant 4$, et qui renvoie la liste des entiers de Carmichael compris entre a et b.

Déterminer à l'aide de cette fonction la liste des entiers de Carmichael inférieurs à 10 000.