DM 3 : TROIS PROBLÈMES

À rendre pour le dimanche 24 mars 23h59.

Ce DM comporte trois problèmes à faire sur ordinateur (Pyzo, Basthon, etc.). Vous devez écrire un fichier qui contient un script python au format **.py** et l'envoyer par mail.

Vous devez impérativement essayer de faire au moins un problème parmi les trois (celui de votre choix). Tentez d'abord de le faire par vous-même le plus possible, puis si vous n'y arrivez pas faites-vous aider sans pour autant regarder un script « tout fait ». Bien entendu, vous pouvez tout à fait traiter les trois problèmes pour tenter de décrocher un triple A !

Dans la suite, étant donné une fonction `f` d'argument `x`, on écrira `f(x) = a` pour signifier que l'instruction `f(x)` doit renvoyer la valeur `a` (ce n'est pas une affectation).

1 Entiers SFC

Un entier naturel $n > 0$ est dit *sans facteur carré* (en abrégé SFC) quand aucun des diviseurs de n au moins égaux à 2 n'est un carré. Par exemple, 6 est SFC car ses diviseurs sont 1, 2, 3, 6 et aucun d'entre eux n'est le carré d'un entier au moins égal à 2. En revanche, 12 n'est pas SFC, car 4 divise 12 et 4 est un carré.

1. Écrire une fonction `sfc(n)` de paramètre un entier n au moins égal à 1 et qui renvoie un booléen indiquant si n est SFC ou pas.

   Par exemple, `sfc(1) = True`, `sfc(6) = True` et `sfc(12) = False`.

```
print( sfc(132457) )    # Tests à ajouter au script
print( sfc(13245798) )
print( sfc(123456789) )
```

2. Écrire une fonction `listesfc(n)` de paramètre un entier n et qui renvoie la liste des entiers SFC compris entre 1 et n, rangés dans l'ordre croissant.

   Par exemple, `listesfc(13) = [1,2,3,5,6,7,10,11,13]`

```
print( listesfc(2500)[-1] )   # Test à ajouter au script
```

3. Écrire une fonction `sommesfc(n)` de paramètre un entier n et qui renvoie la somme des entiers SFC compris entre 1 et n.

   Par exemple, `sommesfc(13) = 58`

```
print( sommesfc(1000) )     # Test à ajouter au script
```

4. Écrire une fonction `divsfc(n)` de paramètre un entier n et qui renvoie la liste des entiers k compris entre 1 et n, rangés dans l'ordre croissant, tels que k divise `sommesfc(k)`.

   Par exemple, `divsfc(100) = [1,3,8,37]`

```
print( divsfc(500) )     # Test à ajouter au script
```

2 Rendre la monnaie

En MPSI, il n'y a que deux pièces de monnaie en circulation, de valeurs a et b où a, b sont deux entiers naturels au moins égaux à 2 et premiers entre eux, ce qui rend parfois la vie difficile aux épiciers de la classe. En effet, quand un client tend un billet pour régler ses achats et qu'il faut rendre la monnaie n avec uniquement des pièces de a ou de b, il arrive parfois que ça bloque. Par exemple, si $a = 3$ et $b = 7$, il est impossible de rendre 11 en pièces de 3 ou de 7. En revanche, si $n = 13$, cela devient possible car $13 = 2 \times 3 + 1 \times 7$.

1. Écrire une fonction `monnaie(n,a,b)` de paramètres trois entiers n, a, b et qui renvoie la liste des couples (u, v) d'entiers naturels tels que $n = au + bv$ (on pourra représenter ce couple par une liste à deux éléments), rangés dans l'ordre croissant selon la première coordonnée u.

   Par exemple, `monnaie(11,3,7) = []`, ce qui signifie qu'il n'existe aucune façon d'écrire 11 sous la forme $3u+7v$; `monnaie(34,3,7) = [[2,4],[9,1]]`, ce qui signifie qu'il existe deux façons de rendre la somme 34 avec des pièces de 3 et de 7 : $34 = 2 \times 3 + 4 \times 7$ et $34 = 9 \times 3 + 1 \times 7$.

```
print( monnaie(2024,21,37) )   # test à rajouter au script
```

2. Quand a et b sont premiers entre eux, on peut montrer que si $n > ab$, alors un épicier de la MPSI peut toujours rendre la monnaie en pièces de a ou b. Écrire une fonction `nomoney(a,b)` de paramètres deux entiers a, b et qui renvoie la liste des entiers $p > 1$ rangés par ordre croissant qui ne peuvent pas s'écrire sous la forme $au + bv$ avec u, v entiers naturels.

   Par exemple, `nomoney(3,5) = [1,2,4,7]`

```
print( nomoney(5,7) )   # test à rajouter au script
```

3. Les commerçants de la MPSI ont une manie : ils aiment rendre la monnaie avec le même nombre de pièces de a et de b. Malheureusement, ce n'est pas toujours possible. Alors, quand ils ont le choix, ils rendent la monnaie de telle sorte qu'il y ait à peu près le même nombre de pièces de a que de b.
   Écrire une fonction `moneymaniac(n,a,b)` de paramètres trois entiers n, a, b et qui renvoie le couple $[u, v]$ tel que $|u - v|$ soit minimale parmi les solutions de l'équation $n = au + bv$ s'il en existe. Si cette équation n'a pas de solution, la fonction renvoie une liste vide, s'il y a plusieurs solutions $[u, v]$ qui minimisent $|u - v|$, alors on renvoie celle où u est minimale.
   Pour rappel, la fonction Python `abs(x)` renvoie la valeur absolue d'un nombre.

   Par exemple, `moneymaniac(168,3,7) = [14,18]`, `moneymaniac(8,3,7) = []`.

```
print( moneymaniac(1000,5,7) )   # test à rajouter au script
```

Et maintenant, un peu de culture mathématique... Un mathématicien s'est un jour rendu dans un fast-food qui vendait des McNuggets par paquets de 6, 9 ou 20. Il s'est rendu compte qu'à partir de ces trois tailles, il pouvait commander certains nombres de McNuggets (15, 21, 24...) mais pas d'autres (11, 22, 34...). Plus incroyable encore, à partir de 44 ou plus, il y avait toujours une combinaison de McNuggets qui fonctionne [1] ! Le nombre 43 est appelé **nombre de McNuggets** : c'est le plus grand nombre qu'on ne peut obtenir à partir de boites de McNuggets.

1. Vous pouvez vous amuser à le prouver (ça ne fait pas partie du DM).

3 Sommes de chiffres

Pour résoudre cet exercice, vous n'hésiterez pas à utiliser les fonctions de conversion suivantes :

(i) `str(x)` transforme un objet x en une chaîne de caractères.

(ii) `list(x)` transforme une séquence x en une liste.

(iii) `int(x)` transforme un objet x en un entier si cela a un sens.

Dans tout l'exercice, pour $(n,k) \in \mathbb{N}^2$, on note $s_k(n)$ la somme des chiffres de n^k (sous-entendu : chiffres en base 10, bien sûr).

1. Écrire une fonction `chiffres(n)` de paramètre un entier n et qui renvoie la liste de ses chiffres (un chiffre est ici une chaîne à un seul élément). Par exemple, `chiffres(125)= ['1','2','5']`.

```
print( chiffres(9**9) )   # test à rajouter au script
```

2. Écrire une fonction `sommechiffres(n)` de paramètre un entier n et qui renvoie la somme de ses chiffres. Par exemple, `sommechiffres(125) = 8`.

```
print( sommechiffres(5**5**5) )   # test à rajouter au script
```

3. Écrire une fonction `maxcarre(n)` de paramètre un entier n et qui renvoie une liste à deux éléments $[a, s_2(a)]$ telle que $s_2(a) = \max\{s_2(i) \mid 1 \leq i \leq n\}$. S'il y a plusieurs solutions pour a, la fonction renvoie la liste où a est minimal.
   Par exemple, `maxcarre(100) = [83,31]`.

```
print( maxcarre(1000) )   # test à rajouter au script
```

4. Écrire une fonction `racine(n)` de paramètre un entier n et qui renvoie le plus grand entier p tel que $p^2 \leq n$. Dans la suite, ce nombre est noté $r(n)$.
   Par exemple, `racine(63) = 7`.

```
print( racine(123456789) )  # "Waouh ! Et si je mettais ce p au carré..."
```

5. On admettra que si n et k sont deux entiers non nuls et si $n = s_k(n)$ alors n a au maximum $1 + r(k)$ chiffres. Écrire une fonction `egauxsomme(k)` de paramètres un entier k et qui renvoie la liste des entiers n tels que $n = s_k(n)$ rangés dans l'ordre croissant. Par exemple, `egauxsomme(3) = [1,8,17,18,26,27]`.

```
print( egauxsomme(10) )   # test à rajouter au script
```