En arithmétique :
de parfaits jumeaux sont amiables,
avec persistance…

EXERCICE 1 : Nombres Parfaits

DEFINITION :
Un entier naturel est dit parfait si il est égal à la somme de ses diviseurs propres.
Par exemple : les diviseurs stricts de 6 sont 1, 2 et 3 et 6 = 1 + 2 + 3 donc 6 est un nombre parfait.

1°) Écrire une fonction parfait qui prenne en entrée un nombre entier et qui réponde True si c'est un nombre parfait et False sinon.

2°) Utiliser la fonction précédente pour créer une fonction inf_parfait qui renvoie le nombre et la liste des nombres parfaits inférieurs à une valeur proposée par l'utilisateur.
En déduire la liste des nombres parfaits inférieurs à 1000 :

Liste des nombres parfaits inférieurs à 5000 :

Liste des nombres parfaits inférieurs à 10 000 :

EXERCICE 2 : Nombres premiers jumeaux

DEFINITION :
Deux nombres premiers sont dits jumeaux si l'écart entre eux est 2.
Par exemple 3 et 5 sont des nombres premiers jumeaux.

1°) Écrire une fonction premier qui renvoie True si le nombre donné par l'utilisateur est premier et False sinon.

2°) Utiliser la fonction précédente pour créer une fonction inf_jumeaux qui renvoie la liste et le nombre de nombres premiers jumeaux inférieurs à une valeur proposée par l'utilisateur.

Liste des nombres premiers jumeaux inférieurs à 100 :

Liste des nombres premiers jumeaux inférieurs à 1000 :

EXERCICE 3 : Persistance

Certains problèmes qu'on explique en quelques secondes semblent hors de portée des raisonnements abstraits et des capacités de calcul des plus puissants ordinateurs.
Le problème de la persistance multiplicative des nombres en base 10, d'énoncé très simple, est l'une de ces redoutables énigmes ou le blocage est total.
Considérons un nombre entier positif, par exemple 377.

Multiplions ses chiffres :$3\times7\times7 = 147$.
Opérons de même avec le résultat $1\times4\times7 = 28$.
Recommençons : $2\times8 = 16$.
Encore : $1\times6 = 6$.

Arrive à un nombre d'un seul chiffre, on ne peut plus rien faire : $377 \rightarrow 147 \rightarrow 28 \rightarrow 16 \rightarrow 6$.
Cette suite est la ≪ suite multiplicative ≫ de 377 et la ≪ persistance multiplicative ≫ p de 377 est le nombre de fois qu'il a fallu multiplier les chiffres avant d'arriver à un nombre à un seul chiffre ; ici, $p = 4$.
Maintenant se posent les questions naturelles et élémentaires suivantes :
– La persistance d'un nombre peut-elle être n'importe quel entier ?
– Si ce n'est pas le cas, quelle est la persistance maximale d'un nombre ?
Ces questions paraissent faciles, dans la mesure où l'ordinateur peut suppléer à nos capacités limitées de manipulation des chiffres.
L'expérience montre qu'il n'en est rien : jusqu'à présent, ni le raisonnement ni le calcul par ordinateur n'ont permis de trouver d'entiers ayant une persistance supérieure à 11 ! Et l'on désespère d'en découvrir, ainsi que de réussir à prouver que 11 est le maximum.

Jean-Paul Delahaye (Pour la science n°430 – Aout 2013).

1°) Écrire une fonction produit qui prend en entrée un entier et qui renvoie le produit de ses chiffres.

2°) En déduire une nouvelle fonction : persistance, qui prend en entier un nombre et qui renvoie sa persistance.
Déterminer la persistance des entiers ci-dessous :

Nombre	10	1235	6788	2677889	26882587	377888999	277777788888899
persistance							

3°) a) Adapter le programme précédent afin de créer une fonction inv_persistance qui renvoie le plus petit entier ayant la persistance demandée par l'utilisateur.
L'utiliser pour compléter le tableau ci-dessous :

Persistance	2	3	4	5	6	7	8
Nombre							

Attention pour une persistance de 8 le temps de calcul peut être long.

b) Créer une fonction affiche_persistance qui renvoie les entiers compris entre 1 et une valeur proposée par l'utilisateur ayant une persistance proposée par l'utilisateur. On créera ainsi une fonction ayant deux paramètres. La fonction devra non seulement afficher les nombres mais aussi afficher le nombre de solutions trouvées.

EXERCICE 4 : Nombres amiables

DEFINITION :
Deux nombres sont dits amiables si chaque nombre est égal à la somme des diviseurs propres de l'autre.

1°) Créer une fonction somme_div qui renvoie la somme des diviseurs stricts d'un nombre.

2°) Créer une fonction amiable qui renvoie le nombre de nombres amiables ainsi que la liste de ces nombres inférieurs ou égaux à une valeur saisie par l'utilisateur.
Liste des nombres amiables inférieurs à 1000 :

Liste des nombres amiables inférieurs à 10 000 :