

```
// Les algorithmes suivants sont destinés aux élèves de CM2 "mordus" au langage LOGO . Difficulté : +++++ //

// Problème n°1 : trouve un nombre dont le reste de la division par 2, par 3 par 5 et par 7 donne 1. Y en a t-il d'autres ?
Solution en deux temps:
1) Faire construire par LOGO les listes de nombres non-multiples par 2, 3, 5 et 7.
( Pour la division par 2, on aura U(n)=( 2* n ) + 1 avec n= 1, 2, 3 ... )
2) Parcourir la liste la plus courte (ici la liste des non-multiples de 7) et rechercher pour chaque élément de la liste s'il
ne figure pas dans les trois autres listes. Si c'est le cas, retenir cet élément pour affichage.
//
```

```
efftxt
donne "liste2 [] donne "liste3 [] donne "liste5 [] donne "liste7 []
ecl [ Construction des listes de nombres des non-multiples. ]
répète 300 [ // nombre à modifier selon l'envie (et le temps ! ) //
donne "liste2 ph :liste2 (2* boucle) +1
donne "liste3 ph :liste3 (3* boucle) +1
donne "liste5 ph :liste5 (5* boucle) +1
donne "liste7 ph :liste7 (7* boucle) +1
]
donne "indice 1 donne "max item :liste2 card :liste2
ecl [ Recherche des solutions: ]
tantque (item :liste7 :indice) < :max [
si ( membre? item :liste7 :indice :liste2 ) et
( membre? item :liste7 :indice :liste3 ) et
( membre? item :liste7 :indice :liste5 ) ecl item :liste7 :indice
donne "indice :indice + 1
]
```

```
// Les 7 premiers nombres-solutions sont: 211, 421, 631, 841, 1051, 1261 et 1471 .
Pour les obtenir, remplacer la ligne:
répète 300 [
par:
répète 800 [
//
```

// Les deux problèmes suivants doivent être traités dans l'ordre 2a puis 2b, car ils nécessitent tous deux une technique de programmation pour extraire la partie décimale d'un nombre qui est illustrée dans le problème 2a. //

// Problème n°2a : Calcule et affiche le chiffre de s unités de tous les nombres compris entre 10 et 99. (il y a donc 90 résultats). Que remarques-tu ?//

```
pour problème2a
efftxt
donne "nombre 10
tantque :nombre <100 [
ecl (( :nombre / 10 ) - (entier( :nombre / 10 ) ) ) *10
// On divise d'abord le nombre par 10 que l'on soustrait à la partie entière de ce résultat, ce qui a pour effet d'obtenir la
partie décimale (le chiffre des unités). On multiplie ensuite cette partie décimale par 10 pour obtenir le chiffre des
unités en valeur entière.//
donne "nombre :nombre + 1
]
fin
```

// Problème n°2b : trouve tous les nombres inférieurs à 100 qui, multipliés chacun par eux-mêmes, terminent le nombre produit (ex: 25x25 = 625). //

```
pour problème2b
efftxt
répète 100 [
donne "résultat boucle * boucle
teste boucle < 10
sivrai donne "diviseur 10
sifaux donne "diviseur 100
donne "résu ((:résultat / :diviseur)- ((entier (:résultat / :diviseur)))) * :diviseur
si (boucle = :résu) ecl ph boucle :résultat
]
fin
```

problème2b

```
// A savoir:
un nombre est dit automorphe s'il se termine par lui-même quand on l'élève au carré (ou à la puissance 2).
//
```

// Voici deux exemples de problèmes (classiques) en résolution par "tâtonnement numérique". Ces algorithmes procèdent de la même façon que l'élève en se basant sur les contraintes figurant dans chaque énoncé.//

// problème n3 //

efftxt

écrisligne [Je suis un nombre. Le produit de mes trois chiffres est 12. La somme de mes chiffres est 7. Qui suis-je ?]

```

donne "u 0 donne "d 0 donne "c 1
tantque (:c < 9) et ((:c *:d *:u)<>12) et ((:c + :d + :u)<>7) [
  donne "d 0
  si ((:c *:d *:u)=12) et ((:c + :d + :u)=7) écrisligne phrase phrase :c :d :u
  tantque (:d < 9) et ((:c *:d *:u)<>12) et ((:c + :d + :u)<>7) [
    donne "u 0
    si ((:c *:d *:u)=12) et ((:c + :d + :u)=7) écrisligne phrase phrase :c :d :u
    tantque (:u < 9) et ((:c *:d *:u)<>12) et ((:c + :d + :u)<>7) donne "u :u + 1
    si ((:c *:d *:u)=12) et ((:c + :d + :u)=7) écrisligne phrase phrase :c :d :u
    donne "d :d + 1
  ]
  si ((:c *:d *:u)=12) et ((:c + :d + :u)=7) écrisligne phrase phrase :c :d :u
  donne "c :c + 1
]
si ((:c *:d *:u)=12) et ((:c + :d + :u)=7) écrisligne phrase phrase :c :d :u

```

// problème n3bis //

écrisligne [Je suis un nombre. Le produit de mes trois chiffres est 12. La somme de mes chiffres est 8. Qui suis-je ?]

```

donne "u 0 donne "d 0 donne "c 1
tantque (:c < 9) et ((:c *:d *:u)<>12) et ((:c + :d + :u)<>8) [
  donne "d 0
  si ((:c *:d *:u)=12) et ((:c + :d + :u)=8) écrisligne phrase phrase :c :d :u
  tantque (:d < 9) et ((:c *:d *:u)<>12) et ((:c + :d + :u)<>8) [
    donne "u 0
    si ((:c *:d *:u)=12) et ((:c + :d + :u)=8) écrisligne phrase phrase :c :d :u
    tantque (:u < 9) et ((:c *:d *:u)<>12) et ((:c + :d + :u)<>8) donne "u :u + 1
    si ((:c *:d *:u)=12) et ((:c + :d + :u)=8) écrisligne phrase phrase :c :d :u
    donne "d :d + 1
  ]
  si ((:c *:d *:u)=12) et ((:c + :d + :u)=8) écrisligne phrase phrase :c :d :u
  donne "c :c + 1
]
si ((:c *:d *:u)=12) et ((:c + :d + :u)=8) écrisligne phrase phrase :c :d :u

```

// Voici les algorithmes généraux qui résolvent les problèmes n3 et 3bis. La taille du nombre ainsi que le produit et la somme des chiffres sont paramétrables à volonté. Seulement pour les experts ! //

pour calculeproduit

donne "p 1

répète card :listechiffres donne "p :p \* item :listechiffres boucle

rends :p = :produit

fin

pour calculesomme

donne "s 0

répète card :listechiffres donne "s :s + item :listechiffres boucle

rends :s = :somme

fin

efftxt

écrisligne [ nombre de chiffres: ] donne "chiffres lisnombre écrisligne :chiffres effl

donne "nombre puissance 10 :chiffres - 1

écrisligne [ valeur du produit des chiffres: ] donne "produit lisnombre écrisligne :produit effl

écrisligne [ valeur de la somme des chiffres: ] donne "somme lisnombre écrisligne :somme effl

donne "nbreboucles 0

répète :chiffres donne "nbreboucles :nbreboucles + (9 \* puissance 10 boucle -1 )

écrisligne phrase :nbreboucles [ cas à examiner. ]

répète :nbreboucles [

donne "listechiffres sépare :nombre

// On transforme le nombre en liste, chaque chiffre du nombre devient un item. //

si (calculeproduit et calculesomme) écrisligne phrase phrase boucle [:] :nombre

donne "nombre :nombre + 1

]

// On aurait très bien pu stopper l'algorithme au premier nombre-solution rencontré et permuter ensuite ses chiffres pour obtenir les autres nombres-solutions. Qui a trouvé ce nouvel algorithme, nécessairement plus rapide ? //

