Mercredi 16 mars 2005. Maple TD 9.

Exercice 1 : *Suite de Perrin.*
Soit $(P_n)_{n\in\mathbb{N}}$ la suite récurrente d'ordre 3 définie par :

$$\begin{cases} P_0 = 3, \;\; P_1 = 0, \;\; P_2 = 2 \\ \forall n \geq 3, \;\; P_n = P_{n-2} + P_{n-3} \end{cases}$$

1. Programmer cette suite.
2. Que dire des diviseurs de P_2, P_3, P_5, P_7, P_{11}, P_{13} ,P_{17},... ?
3. Vérifier que p divise P_p pour les 200 premiers nombres premiers. (On pourra utiliser un test `if` qui renvoie 1 si la condition est vraie, 0 sinon).

* * * * * * * * * * * * *

Exercice 2 : *Décomposition en facteurs premiers.*
Le but de cet exercice est de construire une procédure `decomp` ayant pour argument un **entier** n et qui renvoie la décomposition

$$n = \prod_{i=1}^{k} p_i^{d_i}$$

sous la forme d'une liste de couples $[[\mathtt{p_1}, \mathtt{d_1}], [\mathtt{p_2}, \mathtt{d_2}], ..., [\mathtt{p_k}, \mathtt{d_k}]]$.
Cette procédure se decompose en trois étapes, données ci-dessous. On pourra programmer ces étapes pour une valeur arbitraire de n, puis les rassembler dans une procédure.

1. (a) A l'aide de la commande *ithprime*, construire la liste L des nombres premiers plus petits que n.
   (b) Extraire de L la liste P des nombres premiers qui divisent n.
   (c) A l'aide d'une boucle *while*, construire la liste des couples $[p_i, d_i]$.
2. Tester la procédure sur 9, 1024, 257, 2310.
   Attention : cette procédure est particulièrement lente... Ne pas tester des valeurs trop grandes de n.

* * * * * * * * * * * * *

Exercice 3 : *Nombres parfaits.*

Un nombre entier $n \in \mathbb{N}$ est dit *parfait* s'il est égal à la somme de ses divieurs stricts.
Le but de cet exercice est de vérifier sur des exemples que si le nombre de Mersenne $M_p = 2^p - 1$ est premier, alors $2^{p-1}M_p$ est parfait.

1. Créer la liste $[[M_p, p]...]$ ou p parcours les 50 premiers nombres premiers.
2. Ne garder que les couples $[M_p, p]$ tels que M_p est premier.
3. A l'aide de la commande *divisors*, vérifier qu'alors $2^{p-1}M_p$ est parfait.
   Ind : On pourra créer une procédure *est_parfait(n)* qui renvoie *true* si n est parfait et $false$ sinon, puis l'appliquer à chaque élément de la liste...
   (Pour utiliser *divisors*, il faut appeler la bibliothèque *numtheory*.)

* * * * * * * * * * * *