Les exercices précédés de (*) sont plus délicats, et peuvent être laissés de coté dans un premier temps.

1 Mise en route

1. Mettre en marche l'ordinateur, lancer Maple, et écouter l'éminent orateur.
2. Exécuter les instructions suivantes, en essayant de deviner le résultat *avant*, et de le comprendre après...

```
>2+2;
>1234^1234;
>5/3+21/10;
>5/3+2.1;
>pi;
>Pi;
>cos(pi);
>cos(Pi);
>cos(3.14);
>cos(3.141592654);
>if 3+3=7 then isprime(1789) else ithprime(1789) fi;
>if 3+3=6 then isprime(1789) else ithprime(1789) fi;
>l:=[]:for i from 1 to 10 do l:=[op(l),i] od;l;
>l:=[]:for i from 1 to 10 do l:=[op(l),i] od:l;
```

3. Calculer :
$$\int_0^1 \frac{\sin x \cos x}{\sin^4 x + \cos^4 x} dx, \qquad \sum_{k=0}^{50} k^4, \qquad \sum_{k=0}^{n} k^4.$$
4. Représenter graphiquement successivement puis simultanément les fonctions sinh, cosh et tanh (*utiliser* `plot`).
5. Représenter la surface d'équation $z = \cos(x^2 + y^2)\mathrm{e}^{-(x^2+y^2)/5}$ (*utiliser* `plot3d`).
6. Montrer que la dérivée 170ème de $x \mapsto x^{169}\mathrm{e}^{-1/x}$ est égale à $\frac{\mathrm{e}^{-1/x}}{x^{171}}$.

2 Entiers, flottants

1. Donner les 50 premières décimales de π.
2. Donner la 1789-ème décimale ("après la virgule") de π.
3. Faire `evalf(3*1/3)` puis `3*evalf(1/3)` ; expliquer.
4. Pour Maple, quel est le premier entier premier ?
5. Déterminer combien d'entiers premiers sont compris entre 100 et 1000. On pourra :
   – faire une boucle et utiliser `isprime` ;
   – tatonner avec `ithprime`.

6. Ecrire une procédure `factorielle` qui fait ce que l'on pense[1]. On donnera une version *récursive* et une version *itérative* (précisions en TD).

3 Listes, ensembles

1. Construire une liste $[1, 2, \ldots, 50]$ constituée des 50 premiers entiers > 0. On utilisera successivement `seq`, $, et une boucle `for ... do ... od`.
2. Construire la liste des 50 premiers entiers premiers, puis la liste des entiers ≤ 500 qui sont premiers. Combien en trouve t-on ?
3. Ecrire une procédure changeant une liste en un ensemble et vice-versa.
4. Ecrire une procédure `maxi` prenant en entrée une liste de réels et retournant le maximum de ces réels. Par exemple, `maxi([1,8,-5]);` retournera 8 comme résultat.
5. Ecrire une fonction `indices_pairs` prenant en entrée une liste L, et retournant la liste constituée des éléments d'indices pairs de L (le deuxième, le quatrième, etc ...).
6. Ecrire une fonction `termes_pairs` prenant en entrée une liste L, et retournant la liste constituée des termes pairs de L (par exemple, `termes_pairs([1,3,-2,1,2,4,2,toto]);` retourne `[-2,2,4,2]`).
7. Ecrire une procédure `appartient` qui prend deux arguments x et L, et qui retourne `true` ou `false`, selon que x est ou non l'un des termes de la liste L.
   - **Première méthode** : avec une boucle.
   - **Seconde méthode** : utiliser astucieusement `nops` (facultatif).
8. Ecrire une procédure `doublon` prenant en entrée une liste, et retournant `true` ou `false`, selon qu'il y a ou non une valeur doublée dans la liste (un terme apparaissant deux fois ou plus).

   Précisons qu'il convient de *retourner* un *booléen true* ou *false*, et non *afficher* une *chaine* `‘true‘` ou `‘false‘`.
   - **Première méthode** : faire `deux boucles imbriquées`.
   - **Seconde méthode** : utiliser `nops` (facultatif).
9. (*) Ecrire des procédures prenant en entrée une liste de réels, et déterminant si les éléments de la liste forment une suite :
   - croissante ;
   - monotone ;
   - de signe (large) constant.
10. Ecrire une procédure `inversion` prenant en argument une liste, et retournant cette même liste en ayant inversé l'ordre des éléments.
11. (DS 98) Un polynôme $P = 3 + 7X^2 - 4X^3 + X^5$ peut être représenté par la liste de ses coefficients (en commençant par le coefficient constant : ici `[3,0,7,-4,0,1]`). La méthode de Hörner consiste à l'évaluer en n sommes et n produits (où n est le degré du polynôme) :

$$P(t) = 3 + t\Big(0 + t\Big(7 + t\big(-4 + t(0 + t.1)\big)\Big)\Big).$$

[1] Pour éviter la question qui revient tous les ans : elle prend en entrée un entier n, et retourne $n!$...

Ecrire une procédure `hörner` prenant en entrée un polynôme (sous la forme de la liste de ses coefficients) et un réel (ou une variable) x, et retournant l'évaluation de ce polynôme en ce réel grâce à la méthode de Hörner. Par exemple, `hörner([1,-1,2,3],-2)` retournera -13, alors que `hörner([1,1,0,1],toto)` retournera $1 + toto + toto^3$.

12. (*) On cherche les solutions dans $\mathbb{N}^2$ de $\begin{cases} x^2 - y^2 = 7344 \\ x \wedge y = 12 \end{cases}$

    $x \wedge y = 12$ **signifie** $pgcd(x, y) = 12$**, et le pgcd se dit gcd in english (*greatest common divisor*). Comme il s'agit de pgcd entier, la fonction Maple est `igcd`.**

    (a) Montrer (sans Maple !) que l'ensemble des solutions est fini.

    (b) Trouver les solutions dans $\mathbb{Z}^2$ de $x^2 - y^2 = 7344$. Pour cela, utilisez `isolve`. Quel est le type du résultat ?

    (c) Ne conserver que les solutions dans $\mathbb{N}^2$. On a accès au membre gauche (resp. droit) d'une équation avec `lhs` (resp. `rhs`) : faites des essais !

    (d) Ne conserver que les solutions vérifiant $x \wedge y = 12$.

4 Exercices plus élaborés

1. Ecrire une procédure `ramanujan`[2] prenant en entrée un entier N, et retournant la suite (ou liste...) des entiers s'écrivant de deux façons différentes comme somme de deux cubes d'entiers $\leq N$. Par exemple, $1729 = 12^3 + 1^3 = 10^3 + 9^3$ est le premier entier concerné. Il doit apparaître dans `ramanujan(12)` mais pas `ramanujan(10)`.

   On pourra chercher les égalités $a^3 + b^3 = c^3 + d^3$ avec a le plus petit de a, b, c, d, et $c \leq d$. Cela reviendra donc à faire 4 boucles imbriquées.

2. Ecrire un programme prenant comme argument un réel x et un entier positif n, et calculant x^n grâce à l'*algorithme d'exponentiation rapide*. Le calcul de x^{23} est par exemple facilité si on note que $23 = 16 + 4 + 2 + 1$, donc $x^{23} = x * x^2 * x^4 * x^{16}$: il suffit de calculer successivement les x^{2^k} (précisions données en TD).

3. Montrer que pour $n \in [\![1, 100]\!]$, la dérivée n-ième de $x \mapsto x^{n-1}\mathrm{e}^{-1/x}$ est égale à $\dfrac{\mathrm{e}^{-1/x}}{x^{n+1}}$.

4. Montrez que les 100 premiers polynômes cyclotomiques[3] ont leurs coefficients dans $\{-1, 0, \}$. Le résultat est-il valable au delà ?

5. Calculer $f \circ f \circ f(4444^{4444})$, où f est l'application qui associe à un entier la somme des chiffres de sa représentation en base 10. Question subsidiaire : retrouver le résultat à la main !

6. Ecrire une procédure `TVI` simulant la dichotomie vue dans la preuve du théorème des valeurs intermédiaires : la procédure prend une fonction f (continue), deux réels a et b tels que $f(a)f(b) < 0$, un réel $\varepsilon > 0$, et renvoie $c \in [a, b]$ tel que $f(x) = 0$ pour un certain $t \in [c - \varepsilon, c + \varepsilon]$. Exemples :

[2] La très belle origine historique du nom de cette procédure vous sera expliquée en TD...
[3] Vous ne savez pas ce qu'est un polynôme cyclotomique ? Maple oui !

```
> TVI(x->x^2-2,1,2,.0001),TVI(sin,3,4,.000001);
                              1.414245606, 3.141592983
> evalf(sqrt(2)),evalf(Pi);
                              1.414213562, 3.141592654
```

7. Ecrire une procédure `tri` prenant en entrée une liste de réels, et retournant cette liste triée par ordre décroissant des éléments.

   On décrira AVANT comment va opérer la procédure, puis on évaluera "à la louche" le nombre de comparaisons effectuées pour trier un tableau de n réels.

8. Donner la liste des solutions du problème suivant : on cherche à remplir les 8 cases vides du tableau suivant, sachant que les entiers recherchés sont tous distincts et dans $[\![1, 16]\!]$:

$$\begin{array}{ccccccc} \cdot & + & \cdot & - & \cdot & = & 12 \\ + & & + & & + & & \\ 17 & + & \cdot & - & \cdot & = & 16 \\ - & & - & & - & & \\ \cdot & + & \cdot & - & \cdot & = & 11 \\ = & & = & & = & & \\ 18 & & 8 & & 9 & & \end{array}$$

   Une solution sera sous la forme d'une liste de 9 entiers, comme par exemple : `[2,13,3,17,9,10,1,14,4]`. Combien trouvez-vous de solutions ?

   On a (fortement) intérèt à chercher à résoudre le problème la main, pour déterminer l'algorithme que l'on va programmer.

9. (*) Numération de Fibonacci :

   (a) $(F_n)_{n\in\mathbb{N}}$ désigne la suite de Fibonacci, définie par $F_0 = 0$, $F_1 = 1$, et $F_{n+2} = F_n + F_{n+1}$ pour tout $n \in \mathbb{N}$. Montrer que tout entier $n\geq 2$ s'écrit de façon unique $F_{n_k} + F_{n_{k-1}} + \cdots + F_{n_0}$, où $n_k \geq n_{k-1} + 2$, $n_{k-1} \geq n_{k-2} + 2$, ..., $n_1 \geq n_0 + 2$.

   On pourra raisonner par récurrence et considérer le plus grand entier i tel que $F_i \leq n$.

   (b) Ecrire ue procédure prenant en argument un entier n, et retournant la liste $[n_0 \dots n_k]$ des indices intervenant dans sa décomposition de Fibonacci donnée dans la question précédente.

10. Ecrire une procédure renvoyant le nombre de zéros terminaux dans la représentation décimale de $n!$

    Attention, la procédure ne doit bien entendu pas calculer $n!$...

11. On définit par récurrence les *nombres de Catalan* : $c_0 = 1$, puis $c_n = \displaystyle\sum_{k=0}^{n-1} c_k c_{n-1-k}$ si $n \in \mathbb{N}^*$, de sorte que $c_1 = 1$, $c_2 = 2$ et $c_3 = 5$.

    (a) Ecrire une procédure `catalan1` **récursive** prenant en argument un entier n, et retournant c_n. Quel est le problème de cette procédure ?

    (b) Ecrire une procédure **non récursive** `catalan2` faisant le même travail, en créant, pour le calcul de c_n, un tableau de taille n permettant de stocker petit à petit les caleurs des c_k.