Corrigé des exercices du 20 octobre 2005

1 Quelques commandes utiles

- Message d'erreur:
  Lorsqu'on veut mettre un message d'erreur dans une procécure, on utilise la syntaxe: `error "`*message*`"`. Attention: le message d'erreur est une chaîne de caractères (*string*), pas autre chose!

- Renvoyer du texte:
  Lorsqu'on veut renvoyer du texte, plutôt que d'utiliser une chaîne de caractère "*chaine*" dont on voit toujours les guillemets, on utilise la syntaxe ‘*texte*‘ (Attention, ne pas confondre ‘ avec ’).

- `print`:
  Une procédure Maple nous renvoie toujours le résultat de la dernière commande exécutée mais on a parfois envie de regarder des résutats intermédiaires de ses calculs.
  Pour celà on peut utiliser la syntaxe `print(`$expr1, expr2, ...$`)`.

2 Correction

Exercice 1: polynôme de degré 2

```
second_degre:=proc(a,b,c)
   local delta;
      if a=0 then
          error "a doit etre non nul"
      else
         delta:=b^2-4*a*c;
          if delta<0 then ‘pas de solution‘
             elif delta=0 then ‘racine double ‘,x=-b/(2*a)
             else ‘2 racines distinctes ‘,
 x=(-b-sqrt(delta))/(2*a),x=(-b+sqrt(delta))/(2*a))
          end if;
      end if;
end proc;
```

Exercice 2: somme itérative et récursive

Itératif = boucle for ou $while$
Récursif = le programme s'appelle aux ordres inférieurs. Il faut lui donner une condition d'arrêt, c'est à dire une valeur lorsque l'argument vaut 0 ou 1...
Ecrivez une procédure qui donne le n-ème terme de la suite de Fibonnacci.

- itératif:

```
somme:=proc(n::posint)
   local i,s;
      s:=0;
      for i to n do s:=s+i end do;
end proc;
```

- récursif:

```
somme:=proc(n::posint)
        if n=0 then 0 else n+somme(n-1) end if;
end proc;
```

Exercice 3: renverser les chiffres

```
renverser:=proc(n::posint)
    local a,i,L,n1,n2;
       n2:=0;n1:=n;
        L:=floor(ln(n)/ln(10))+1; # longueur du nombre n
        for i from 0 to L-1 do
            a:=floor(n1/(10^(L-1-i)));
            if a=0 then error "le chiffre 0 n'est pas autorise" else
                n2:=n2+10^i*a;
                n1:=n1-a*10^(L-1-i);
                print(a,n1,n2)
            end if;
        end do;
        n2
end proc;
```

Ici, la longueur L du nombre n est obtenue par l'application du logarithme en base 10 ($\log_{10}(x) = \frac{\ln(x)}{\ln(10)}$). Regardons ce qui se passe dans la boucle for. A chaque étape, on grignote le chiffre de gauche de $n1$ (qui vaut initialement n), appelé a et on l'ajoute à gauche dans $n2$ (qui est le nombre renversé quand le programme est terminé). Pour observer ce qui se passe, on utilise la commande `print`...