Maple TD3

1 Correction des exercices précédents

1.
```
somme1:=proc(n)
local x,i; x:=0;
for i from 0 to n-1 do x:=x+2*i; end do;
return x; end proc;

somme2:=proc(n)
local x,i; x:=0; i:=0;
while i < n do
x:=x+2*i; i:=i+1;
end do;
return x; end proc;
```
2.
```
puissance1:=proc(x,n)
local y,i; y:=1;
for i from 0 to n-1 do y := y*x; end do;
return y; end proc;

puissance2:=proc(x,n)
local y,i; y:=1; i:=0;
while i < n do y:=y*x; i:=i+1; end do;
return y; end proc;

puissance3:=proc(x,n)
if n < 1 then return 0
else return x*puissance3(x,n-1)
end if; end proc;
```

2 Exercices

2.1 Manipulations de séquences

En plus des entiers, des procédures ou des flottants, Maple a un type intégré appelé séquence. Il sert à regrouper des objets dans une sorte de liste. Par exemple `1,2,3,4` est une séquence. Vous pouvez accéder au i^{eme} élément d'une séquence à l'aide de `[i]`. Par exemple : `sequence :=0,1,2,3,4,5 ; sequence[3] ;`. Vous pouvez aussi mettre des séquences bout à bout, en faisant `sequence1,sequence2`.

1. Étudier comment fonctionnent les fonctions `seq` et `nops`
2. Écrire une instruction renvoyant une séquence contenant les n premiers entiers pairs.
3. Écrire une procédure prenant une séquence en paramètre et renvoyant son plus grand élément.
4. Écrire une procédure prenant une séquence en paramètre et renvoyant la somme de ses éléments.
5. Écrire une procédure effectuant la somme des n premiers entiers pairs à l'aide des questions précédentes.

2.2 Dessin élémentaire

1. Étudiez comment fonctionne la fonction `plot`.
2. Représenter la fonction sinus dans l'intervalle $[-2\pi...2\pi]$.

3. Représenter la fonction tangente sur l'intervalle $[-2\pi...2\pi]$. Que remarquez-vous ? Corrigez cet effet en ajoutant l'option `discont=true`.
4. Exécuter `plot([cos(t),sin(t),t=0..2*Pi])` ; Que devriez vous obtenir ? Comment corriger ce problème ?
5. Représenter les fonctions sinus et cosinus sur le même graphe. Comment contrôler la couleur des courbes ?
6. Que fait le code suivant :

```
l:=[0,1]:
for x from 0 to evalf(3*Pi) by 0.2 do
  l:= l,[x,cos(x)]:
end do:
plot([l]);
```

   Le modifier pour écrire une procédure `myplot` prenant en paramètre une fonction f, le début d'un intervalle, la fin d'un intervalle et un pas et traçant elle même le graphe de f sur cet intervalle.

2.3 Triangle de Sierpiński

Voici les premiers termes d'une suite de dessins appelés les triangles de Sierpiński :

C'est un dessin fractal, appelé triangle de Sierpiński. Nous allons implémenter une procédure traçant le triangle à l'itération n.

1. Écrire une séquence `s` telle que `plot(s)` affiche un triange équilatéral.
2. Écrire sur une feuille comment passer d'un triangle au suivant sans tenir compte du grossissement des triangles.
3. Écrire une procédure `translation(p,vx,vy)` prenant en paramètre un point p donné sous la forme d'une séquence de deux réels et les deux coordonnées d'un vecteur v et renvoyant le translaté de p par v.
4. Écrire une procédure `translationSeq(s,vx,vy)` prenant en paramètre une séquence s de points et un vecteur v sous la forme de deux coordonnées et renvoyant la séquence des translatés des points de s suivant v. Utilisez la procedure `translation`
5. Écrire une procédure `suivant(s,n)` prenant une séquence s décrivant le n-ième triangle de sierpinski et renvoyant la séquence décrivant le triangle suivant.
6. Écrire une procédure `sierpinski(n)` utilisant les fonctions précédentes et renvoyant les points du n-ieme triangle de Sierpiński et l'afficher.