ULCO **2018 - 2019**
L1 Informatique **Calcul Formel**

TP 4 – Listes, boucles, tests conditionnels en `Maxima`

1 Manipulation de listes

1.1 Création de listes

Une liste est une expression dans laquelle des éléments en nombre fini sont placés entre deux crochets et séparés par des virgules. Par exemple :

```
(%i1) L:[a,a+2*3*f,8,d];
(L)   [a,6f+a,8,d]
```

On imbrique souvent les listes les unes dans les autres soit parce que la structure du travail s'y prête, soit par exemple parce que l'on manipule des coordonnées de points, etc. Par exemple :

```
(%i1) L2:[[2,1],[5,3],L];
(L2)  [[2,1],[5,3],[a,6f+a,8,d]]
```

La commande `length` renvoie la longueur d'une liste :

```
(%i3) length(L);length(L2);
(%o2)  4
(%o3)  3
```

La fonction makelist permet de créer des listes génériques. On lutilise de deux manières différentes illustrées par les exemples suivants :

```
(%i3) makelist(i^2,i,2,5);
(%o3) [4,9,16,25]
```

```
(%i3) makelist(i^2,i,[2,3,0,-1]);
(%o3) [4,9,0,1]
```

On peut créer des listes d'objets non entiers ou des listes de listes, des listes d'équations, etc... Par exemple :

```
(%i3) L3:makelist(makelist(j,j,1,i),i,1,4);
(L3) [[1],[1,2],[1,2,3],[1,2,3,4]]
```

```
(%i3) equa:makelist(x^i=1,i,1,4)
(equa) [x=1,x^2=1,x^3=1,x^4=1]
```

1.2 Eléments d'une liste

On accède à un élément dune liste en plaçant entre crochets son indice à côté du nom de la liste. Par exemple :

```
(%i55) L:[2,7,-3,toto]$ L[1]; L[length(L)];
(%o54) 2
(%o55) toto
```

Quand il y a plusieurs listes imbriquées, on procède naturellement :

```
(%i98) L:[[2,3],[-1,4],toto,[tutu]]$
L[1]; L[2];
L[1][2]; L[4][1];
(%o95) [2,3]
(%o96) [-1,4]
(%o97) 3
(%o98) tutu
```

1.3 Opérations sur les listes

La fonction `append` permet de concaténer les élements de plusieurs listes en une seule liste :

```
(%i106) L:[2,3];append(L,[5,5]);
(L) [2,3]
(%o106) [2,3,5,5]
```

Les commandes `cons` et `endcons` permettent de placer un élement à la première et à la dernière place d'une liste :

```
(%i111)cons(tyty,L);
       endcons(tyty,L);
(%o110) [tyty,2,3]
(%o111) [2,3,tyty]
```

Vous trouverez dans l'aide de nombreuses commandes sur les listes. Par exemple:

1. La commande `member` qui permet de savoir si un objet est membre d'une liste.
2. La commande `delete` pour supprimer un élément de la liste.
3. La commande `reverse` qui retourne les éléments de la liste.
4. La commande `sort` qui trie les éléments de la liste.
5. La commande `lmax` qui retourne le plus grand des éléments d'une liste.
6. La commande `flatten` qui supprime les crochets délimitants les listes à une liste
7. Ecetera.

2 Les boucles et les tests conditionnels

2.1 Les tests conditionnels

Un test obtenu avec la commande `if` réalise une succession de commandes si la condition placée après `if` est vérifiée. Dans le cas contraire et si on le souhaite, le test réalisera une autre succession de commandes après l'opérateur `else`.

Ces successions de commandes alignées les unes après les autres, séparées par des virgules et entourées de parenthèses constituent la forme la plus simple de "procédure" en `Maxima`.

```
(%i22)  a:10$
        if a<10  then (a:a+2,b:5*a) else (a:a-2,b:a)
(%o22) 8
```

Il ne doit en aucun cas y avoir un point-virgule avant l'opérateur `else`. Même dans l'exemple trivial suivant :

```
(%i22) a:3.14159$
          if a<%pi  then print("plus petit") else print("trop grand")
(a) trop petit
```

Revenons un instant sur ces fonctions élémentaires: leur valeur finale est le dernier objet évalué juste avant la parenthèse finale. Lisez et testez avec soin l'exemple suivant :

```
(%i23) toto:(a:3,b:2^a,c:a+b);
       toto^2;
(toto) 11
(%o23) 121
```

2.2 Les boucles

Donnons par l'exemple différentes formes d'une boucle en Maxima. La liste ne prétend pas être exhaustive. Observer et comprendre ce que donne `Maxima` quand on effectue les boucles suivantes :

1. ```
   (%i1) for i:3 thru 7 do display(i);
   ```

2. N'abusez pas de la commande `display` au sein des boucles ; elle est juste utilisée dans ce paragraphe pour vous montrer qu'elle existe et visualiser le "fonctionnement " des boucles dans ces exemples.

   ```
   (%i55) S:0;
   for i:2 step 2 thru 10   do (S:S+i,display(S));
   ```
   ```

3. Une boucle avec juste l'opérateur `while`.

```
(%i55) k:2;
       while k<100 do
       (
          k:k+3,
          if mod(k,13)=0 then  printf(true,"l'entier ~d est
                                                divisible par 13~%",k)
       );
       k;
```

`Ci-dessus, la présence de ~% permet de passer à la ligne dans ce "print formated".`

4. Ci-dessous on crée la liste des entiers k entre 1 et 100 tels que l'entier $k^2 + 5$ est divisible par 7. Prenez le temps de bien comprendre ce qui se passe avec l'affectation `L:endcons(k,L)`.

   De plus, n'oubliez pas d'initialiser la liste avant d'entamer ces modifications itératives.

```
(%i55) L:[];
for k:1 thru 100 do
if mod(k^2+5,7)=0 then L:endcons(k,L);
L;
```

Exercice 1. La suite de Fibonacci est déterminée par ses deux premiers termes $u_0 = 0$, $u_1 = 1$ et la relation de récurrence $u_n = u_{n-1} + u_{n-2}$ définie pour $n \geq 2$.

Ecrire un programme non récursif qui à partir d'un entier n fixé (pourquoi pas $n = 10$), affiche successivement les valeurs des termes de la suite jusque u_n.

Copier/Coller ensuite votre programme et modifiez-le pour qu'il donne les valeurs des termes de la suite au sein d'une liste et terminer en affichant cette liste.

Exercice 2. Conjecture de Syracuse
On choisit un entier n. Si n est pair on le remplace par $n/2$, sinon on le remplace par $3n + 1$, et ainsi de suite tant que le nombre obtenu est différent de 1. Par exemple, partons de $n = 6$: la suite des entiers générée par le processus est $6, 3, 10, 5, 16, 8, 4, 2, 1$.

Collatz a conjecturé que pour tout entier n, on aboutit toujours à 1. C'est un problème ouvert encore aujourd'hui, c'est à dire que l'on ne sait pas le démontrer.
[Wikipédia : Cette conjecture mobilisa tant les mathématiciens durant les années 1960, en pleine guerre froide, qu'une plaisanterie courut selon laquelle ce problème faisait partie d'un complot soviétique visant à ralentir la recherche américaine].

Ecrire un programme avec `Maxima` qui construit une liste contenant la suite des entiers générée par le processus, puis affiche à l'écran la liste.

Exercice 3. Un tracé d'un polygone régulier

Les points d'un polygone régulier à $n+1$ sommets et placés sur le cercle unité peuvent être obtenus par $M_k = (\cos(2k\pi/n), \sin(2k\pi/n))$ ceci pour $k = 0, \ldots, n-1$.

Choisir un entier n raisonnable et définir avec une commande `makelist`, une liste `tt` contenant les coordonnées des points M_k (les coordonnées d'un point seront ici entourées de crochets et non pas de parenthèses). Obtenir un dessin par une commande `draw2d`.

Exercice 4. Certains d'entre vous démontreront l'année prochaine que

$$\lim_{n\to+\infty}\left(1+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{n^2}\right)=\frac{\pi^2}{6}$$

Comme la suite de terme général $v_n = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{n^2}$ est croissante, cette convergence donne que tous les termes de la suite sont dans l'intervalle $[\frac{\pi^2}{6} - 0.1, \frac{\pi^2}{6}]$ à partir d'un rang N. On peut se représenter le réel 0.1 comme la précision avec laquelle on approxime ici $\frac{\pi^2}{6}$.
Programmer à l'aide d'un `while` une détermination du rang N. Refaire tourner votre boucle pour une précision 0.01, 0.001. Voyez-vous apparaitre quelque chose?