TP Cryptographie
Codage Numérique

NOMS et Prénoms des étudiants :
Groupe :

1 - Initiation à Maple.

Maple est un logiciel de calcul formel. Il permet d'effectuer de très nombreux calculs numériques et surtout symboliques. Ses possibilités graphiques sont également très étendues, et il possède un langage de programmation complet intégré.

Sur une **feuille de calcul** de Maple (comme celle-ci), on présente les calculs à peu près de la même façon que sur une feuille de papier, en tenant compte de quelques particularités syntaxiques du logiciel : par exemple on écrit "Pi" (pour la constante 3, 14...), on note exp(x) (pour l'exponentielle), etc....

****
Dans tous les exemples qui suivent, il vous est demandé :

1 - De comprendre la signification de la commande utilisée.
2 - D'observer attentivement la syntaxe de la commande utilisée, car vous en aurez besoin par la suite.
3 - D'exécuter la commande.
4 - D'observer attentivement le résultat affiché.
5 - Si vous ne comprenez pas l'écriture du résultat affiché ou sa signification, de vous aider de l'aide en ligne (voir exercice 1 ci-dessous) et/ou de poser la question à l'enseignant (cela fait aussi partie du TP).
****

Pour écrire une commande, on doit se placer après le prompt ">" et terminer la ligne par ";".
Pour exécuter une commande (c.-à-d. pour effectuer un calcul), on doit se positionner sur la ligne et appuyer sur la touche "Entrée".
Le résultat s'affiche en dessous.

Exemple :

```
> 45+13;
```

Si on oublie le point-virgule, on obtient un avertissement, et Maple propose une

correction :

```
> 15+16
```

Remarque : On peut aussi terminer une commande par ":" au lieu de ";". Dans ce cas, quand on appuie sur la touche "Entrée", le calcul est effectué, mais pas affiché. À utiliser en cas de calculs intermédiaires lourds et sans intérêt.

Exemple :

```
> 3+5;
  556284*6555843651:
  4*3;
```

a - Calcul.

La syntaxe des opérations élémentaires est la suivante :
Addition : x+y
Multiplication : x*y Attention : symbole * obligatoire !
Soustraction : x-y
Division : x/y
Puissance : x^y Attention : symbole ^ obligatoire ! On écrit "a^2" pour le carré de a.
Racine carrée : sqrt(x)
Exponentielle : exp(x)
Logarithme : ln(x)

Attention : Maple distingue les lettres majuscules et les minuscules. Les commandes "exp" et "Exp" sont donc distinctes.

Nombres rationnels, nombres réels, valeurs approchées
Par défaut, Maple effectue des calculs symboliques : autrement dit, il travaille avec les valeurs exactes des nombres, représentées pas leur écriture symbolique (comme 1/3 ou $\sqrt{2}$), et pas avec des valeurs approchées représentées en virgule flottante (comme 0,333333 ou 1,414213562). Pour obtenir des valeurs numériques approchées, on utilise la commande "evalf" (pour EVALuer en Flottant).

Exemples :

```
> 1/2+2/3+3/4;
> 1/(sqrt(2)-1);
> exp(5);
> ((((((((((((sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt
  (sqrt(sqrt(3)))))))))))))^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2);
> evalf(1/(sqrt(2)-1));
> evalf(exp(5));
> evalf(exp(5),50);
```

Exercice 1 : Effectuer quelques calculs simples puis plus compliqués. N'hésitez pas à être ambitieux, la puissance de calcul de Maple est vraiment grande.

Utilisation de l'aide :
Pour obtenir de l'aide sur une fonction, il suffit de taper son nom précédé d'un point d'interrogation. On peut aussi se servir du menu "Help". Dans les deux cas, on obtient une fenêtre d'aide qui détaille (en anglais...) le fonctionnement de la commande (syntaxe, paramètres nécessaires, type de valeur retournée...) et donne des exemples. N'hésitez pas à l'utiliser. En particulier, il est souvent très pratique de "Copier-Coller" les exemples dans la feuille de calcul.

Utilisation du mode texte :
Pour expliquer et commenter vos hypothèses, vos calculs et vos résultats, vous pouvez insérer du texte dans une feuille de calcul en utilisant le bouton "T" de la barre d'outils.

b - Variables.

Une variable en Maple est décrite par un nom commençant par une lettre.

L'assignation des variables se fait grâce au symbole "**:=**" (et pas "=" comme en langage C, attention !)

```
> x:=3;
> x;
> x+2;
```

On peut assigner n'importe quelle expression mathématique à une variable :

```
> expr:=u^2+sin(v);
> expr-u^2;
```

Important : Pour libérer une variable, il suffit de lui assigner son propre nom entre apostrophes.

Par exemple, actuellement *x* et *expr* sont deux variables qui contiennent des valeurs :

```
> x;
  expr;
```

Mais on peut décider de libérer *x* par cette méthode :

```
> x:='x';
```

Et si on demande à nouveau quel est leur contenu, on voit que *x* a été vidée :

```
> x;
  expr;
```

On peut aussi libérer **toutes** les variables grâce à la commande "restart" :

```
> expr;
```

```
> x:=58;
> restart;
> expr;
  x;
```

Remarque : Certains noms de variables sont réservés à des constantes et ne peuvent être modifiés.
Par exemple Pi est égal à 3.14159....

```
> evalf(Pi);
> cos(Pi);
> Pi:=4;
```

Remarque : Vous pouvez constater que les messages d'erreur de Maple sont écrits en rose...

Important : Un bon réflexe à avoir est de commencer tout nouvel exercice par un "restart" dans le but de purger la mémoire de Maple et de ne pas avoir des variables assignées intempestives. Par exemple si dans l'exercice 1, on a assigné "x:=5", et que dans l'exercice 2 on s'intéresse au polynôme P=x^2+3*x+1, on a alors P=5^2+3*5+1=41 !! Maple "ne sait pas" tout seul qu'on a changé d'exercice ...

2 - Quelques commandes utiles.

On donne dans cette partie les principales commandes dont vous aurez besoin dans ce TP.

```
> restart;
```

a - Les listes.

Pour créer une liste *u* d'éléments numérotés à partir de 1, on utilise la commande "seq" (car "sequence" signifie suite en Anglais), et on met la commande entre crochets "[...]" pour que Maple reconnaisse la structure de données "liste" :

```
> u:=[seq(i^2,i=1..9)];
```

Pour faire référence à un élément d'une liste, on met l'indice entre crochets :

```
> u[4];
  u[1];
  u[20];
```

Pour calculer la somme des éléments d'une liste entre deux indices, on utilise la

commande "add" :

```
> S:=add(u[i],i=1..9);
```

Vérification :

```
> 1+4+9+16+25+36+49+64+81;
```

b - Autres commandes.

On calcule le quotient et le reste d'une division euclidienne à l'aide des commandes "iquo" (pour Integer QUOtient) et "irem" (pour Integer REMainder). Par exemple, le quotient de la division euclidienne de 16 par 3 est 5, et le reste est 1 :

```
> iquo(16,3);
> irem(16,3);
> irem(123456789,10^6);
> iquo(123456789,10^6);
```

Vous aurez aussi besoin de calculer le plus petit nombre entier supérieur ou égal à un certain réel. Par exemple, pour 5.268, on doit obtenir 6. On utilise alors la commande "ceil" :

```
> ceil(5.268);
```

Remarque : en anglais "ceiling" signifie "plafond", ce qui constitue un moyen de se souvenir de cette commande.

Enfin, la commande "length" calcule la longueur de ce qui est donnée en entrée (vu comme une chaîne de caractères) :

```
> length(248);
  length(u);
  length(S);
  length(y);
```

Exercice 2.

Question 1 - Créer la liste v dont les éléments valent 1560+2^i pour i compris entre 1 et 13.

On remarque que tous les nombres de la liste v comportent 4 chiffres en base 10.

On veut se servir des v[i] pour écrire le nombre N= 975256563608258420721816168816241592157615681564156 2 obtenu en collant bout à bout les valeurs v[13]v[12]...v[2]v[1] (dans cet ordre).

Question 2 - Par quelle puissance de 10 doit-on multiplier v[i] pour qu'il apparaisse à la bonne place dans N ?

Question 3 - Créer la liste w qui contient ces puissances de 10.

Question 4 - Calculer N en utilisant v et w.

Dans les trois questions suivantes, on veut découper N en paquets de 7 chiffres, en commençant par la droite. Par exemple, le troisième paquet est : 4159215.

Question 5 - Combien y-a-t-il de paquets ? Indication : on pourra utiliser la commande "length" et la commande "ceil".

Question 6 - Calculer le troisième paquet de 7 chiffres de N en partant de la droite. Indication : on pourra utiliser le quotient de la division euclidienne de N par une puissance de 10 bien choisie, et le reste de la division euclidienne du nombre obtenu par une autre puissance de 10.

Question 7 - Créer la liste x contenant les paquets de 7 chiffres de N.

3 - Exercice 3 : Conversion d'un texte en message numérique et réciproquement.

La première étape avant de faire de la cryptographie consiste à convertir le texte qu'on veut transmettre secrètement en un message numérique, c'est-à-dire en une liste de nombres.
Ensuite, on effectue les calculs nécessaires pour crypter ce message numérique. La

liste des nombres cryptés s'appelle un cryptogramme. C'est ce cryptogramme qui est transmis.

A l'inverse, quand on reçoit un cryptogramme, on effectue les calculs nécessaires pour le décrypter. On obtient alors un message numérique.
La dernière étape du processus consiste à convertir ce message numérique en un texte pour pouvoir le lire.

Dans cette partie, on s'occupe des étapes de conversion d'un texte en message numérique avant le cryptage et de conversion d'un message numérique en texte après le décryptage. Cette méthode sera réutilisée dans les TP suivants.

a - Conversion d'un texte en message numérique.

```
> restart;
```

Supposons qu'on veuille envoyer secrètement le texte suivant, donné sous la forme d'une chaîne de caractères :

```
> texte:="Il y a de nombreuses méthodes possibles pour
  transformer un texte en message numérique. On va utiliser la
  méthode de l'exercice 2. Vous allez partir de la liste des
  codes ASCII (en décimal) des caractères du texte. Dans un
  premier temps, vous devrez les coller les uns à côté des autres
  pour obtenir un TRES GRAND nombre. Ensuite, vous devrez
  découper ce nombre en une liste de blocs de chiffres d'une
  taille plus raisonnable, qui sera le message numérique.";
```

Question 1 - Calculer le nombre de caractères du texte. Ranger le résultat dans une variable appelée "nombrecar".

Conversion texte/message Etape 1/3 - Pour commencer, on calcule la liste des codes ASCII des caractères du texte. Il existe pour cela une commande Maple :

```
> listeascii:=convert(texte,bytes);
```

Ce qu'on obtient est déjà un message numérique, puisque c'est une liste de nombres. Mais il est trop simple, car les nombres de la liste sont petits. Si on cryptait directement ce message numérique, il serait trop facile à décrypter. On doit se ramener à un message un peu plus compliqué.

Conversion texte/message Etape 2/3 - Le code ASCII d'un caractère est un nombre compris entre 1 et 255, il comporte donc au maximum trois chiffres en base 10. Pour savoir où commence et où finit chaque code ASCII, on suppose qu'ils comportent tous 3 chiffres en base 10, autrement dit on utilise 046, 097, etc.

Question 2 - Calculer le nombre obtenu en collant les nombres de la liste "listeascii" les uns à côté des autres. Ranger le résultat dans une variable appelée "nombre".

Attention *: le premier nombre de la liste correspond aux chiffres de poids faibles, et le dernier nombre de la liste correspond aux chiffres de poids forts, comme dans l'exercice 2.*

Question 3 - Calculer le nombre de chiffres de ce nombre. Ranger le résultat dans une variable appelée "nombrechif".

Conversion texte/nombre Etape 3/3 - Dès qu'on veut transmettre un message de quelques lignes, on obtient des nombres difficiles à manipuler, car trop grands. Pour remédier à cela, on redécoupe ce nombre en une liste de plusieurs nombres de longueur plus commode. Par exemple, ici, on découpe le nombre en blocs de 20 chiffres.

```
> longbloc:=20;
```

Question 4 - Calculer le nombre de blocs. Ranger le résultat dans une variable appelée "nombrebloc".

Question 5 - Effectuer le découpage en blocs du nombre qui nous intéresse. Ranger le résultat dans une variable appelée "message".

Remarque : Certains des blocs (en particulier le dernier) peuvent comporter moins de 20 chiffres, car il se peut que certains commencent par des 0.

Le message est maintenant prêt à être crypté et envoyé.

b - Conversion d'un message numérique en texte.

Inversement, supposons qu'on reçoive un cryptogramme qui, une fois décrypté, nous donne ceci :

```
> messagebis := [11708003203303211111809711406 6,
  11511711111803210111711311510 5, 11511723311403212210111809703 2,
  11011109910111403222403210511 5, 10109903211410111711610511611 5,
  11203222403210111612010111603 2, 10903211710003211410511611409 7,
  10911711003210110309711511510 1, 11011110003210111711310511423 3,
  97112233100032117097032233110, 32116115101039099032044116114,
  23403211511711111803210111711 3, 32115116234114112032115101116,
  11611609703211511711111803222 4, 97108032224032114101117113097,
  11411803210110511611409711203 2, 11210910503211611010110910509 7,
```

```
101100032101116110097116114111, 111116112121114099032097108032,
224032044101105104112097114103, 101108032114105111118097115032,
101103097114102102105104099032, 971141021021051040992331000047,
115101109032115101100032101103, 233109117110032115101103097115,
970760320461151011171131051l4, 112032101110105097109101115032,
320441011101050971040991111l4, 100032116101106117115032101108,
320971141011150320800840321l7, 101100111104116233109032097108,
971080321161010320650830820321, 115032101110105097109101115032,
108032044101116110097118105117, 321011001111041162331090320971,
100045224045099097115032117100, 981171110390780320460321151ll,
100032115097112032122101105108, 321141011151051182331140321ll,
109032120117101100032115101099, 111112032115101100111104116233,
114105111118117111112032114117, 101108108105097118097114116032,
101099097099105102102101032114, 33032116110101109];
```

Conversion nombre/texte Etape 1/3 - On commence par recoller les blocs pour former un seul nombre.

On calcule le nombre de chiffres de chaque bloc. Comme certains blocs peuvent commencer par des 0, on cherche la longueur maximale des blocs en utilisant la commande "max", et on espère que tous les blocs ne commencent pas par des 0... On calcule aussi le nombre de blocs du message avec la commande "nops" (ce qui signifie "nombre d'opérandes"...). Ces deux calculs sont effectués à l'aide des commandes ci-dessous :

```
> longblocbis:=max(seq(length(messagebis[i]),i=1..nops
  (messagebis)));
  nombreblocbis:=nops(messagebis);
```

Question 6 - Recoller les blocs de chiffres pour former un seul nombre. Ranger le résultat dans une variable appelée "nombrebis".

Attention *: En cohérence avec le reste du TP, on doit fabrique le nombre 33032 ... 4066, et pas le contraire*

Question 7 - Calculer le nombre de chiffres du nombre obtenu. Ranger le résultat dans une variable appelée "nombrechifbis".

Conversion nombre/texte Etape 2/3 - On reconstitue la liste des codes ASCII des caractères du texte de départ.

Question 8 - Calculer le nombre de blocs de 3 chiffres qu'on obtiendra. Ranger le résultat dans une variable appelée "nombrecarbis".

Question 9 - Découper le nombre obtenu en une liste de blocs de 3 chiffres. Ranger le résultat dans une variable appelée "listeasciibis".

Conversion nombre/texte Etape 3/3 - Chaque bloc est un nombre compris entre 1 et 255, par conséquent c'est le code ASCII d'un caractère. On obtient le texte correspondant par la même commande qu'au début (elle marche dans les deux sens).

```
> convert(listeasciibis,bytes);
```

Pour la conversion dans ce sens, on est obligé de supposer que le nombre qu'on reçoit correspond à un message valide, c'est à dire qu'on obtient des nombres qui sont vraiment des codes ASCII, autrement dit des nombres entre 1 et 255. Pour s'en assurer, on pourrait (mais on ne le fera pas) passer avant la conversion par une étape supplémentaire de détection/correction d'erreurs...