PCSI - exercices d'informatique

Fonctions et procédures

Exercice 1

La séquence d'instructions suivante est-elle correcte ?
> u(0) := 1 ;
> u(n) := u(n-1) + 1/u(n-1)^2 ;

Exercice 2

La définition suivante est-elle correcte ?
> f:=(a,b) -> if (a=0 or b=0) then 0 else 1+f(a-1,b-1) fi;
Que fait-elle ?

Exercice 3

Définir une fonction factorielle(n) donnant :

- la factorielle d'un entier ;
- la factorielle de la partie entière d'un *numeric* ;
- la factorielle de la partie entière d'une valeur approchée d'un *realcons* ;
- une expression non évaluée dans tous les autres cas.

Exercice 4

Définir une fonction maximum() prenant un nombre quelconque d'arguments et retournant le plus grand d'entre eux, en respectant les conditions suivantes :

1. Si un argument au moins n'est pas de type *numeric*, la fonction utilise pour ses calculs des valeurs approchées de ceux-ci (evalf) mais retourne le "vrai" maximum (non approché).
2. Si un argument au moins n'est pas de type *realcons*, la fonction retourne une expression non évaluée.

Exercice 5 *Algorithme d'exponentiation rapide*

Supposons que *Maple*® ne dispose pas de la fonction puissance (^). Il est simple (mais fastidieux) d'élever un nombre x à la puissance n ($n \in \mathbb{N}$) en le multipliant par lui-même $n-1$ fois.

1. Écrire une procédure expoBete(x,n) faisant ce travail.

Toutefois, pour élever x à la puissance, p. ex., 16 ($= 2^4$) il est bien plus judicieux de procéder ainsi : multiplier x par lui-même (x^2), puis le résultat par lui-même (x^4), encore multiplié par lui-même (x^8) et enfin une dernière fois (x^{16}). Si l'on cherchait x^{21} il aurait suffi de multiplier x^{16} par x^4 (calculé en cours de route) et enfin par x.

2. Écrire une procédure expoRapide(x,n) tirant parti de ces remarques.
3. Équiper les procédures précédentes d'un *compteur* de multiplications afin de comparer leurs efficacités.

Exercice 6 *Conjecture de* Collatz

La suite de Collatz est définie par $u_0 = p \in \mathbb{N}^*$ entier (non nul) quelconque et

$$u_{n+1} = \begin{cases} 3u_n + 1 & \text{si } u_n \text{ impair} \\ \frac{u_n}{2} & \text{si } u_n \text{ pair} \end{cases}$$

Par exemple :

$$13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$
$$... \to 4 \to 2 \to 1 \to 4 \to 2 \to 1...$$

La conjecture de Collatz consiste à affirmer que cette suite finira toujours par boucler sur les valeurs 1, 4 et 2, quel que soit l'entier p initial.

1. Ecrire une fonction dureeDeVol associant à p le nombre (dit "durée de vol") d'étapes nécessaires pour revenir à 1 (ainsi dureeDeVol(13) devra retourner la valeur 9).
2. Adapter les instructions précédentes pour obtenir une fonction DVEnAltitude associant à p la "durée de vol en altitude", càd le nombre d'étapes pendant lequel $u_n > p$. (Par exemple DVEnAltitude(13) donnera 2).
3. Écrire une fonction dureeRecord prenant un argument p et retournant sous forme d'une liste les entiers k entre 1 et p présentant une durée de vol "record", càd *plus grande que celle de tous leurs prédécesseurs.*
4. Même question pour la durée de vol en altitude record.

Exercice 7 *Algorithme "reverse-and-add"*

Soit un nombre n, par exemple 96. On retourne n pour obtenir 69 et on ajoute : 165. On recommence : $165 + 561 = 726$, $726 + 627 = 1353$, $1353 + 3531 = 4884$. Ce dernier nombre est un *palindrome*, càd qu'il se lit de manière identique dans les deux sens.

La plupart des nombres entiers initiaux (mais pas tous !) conduisent à un palindrome au bout d'un nombre fini d'étapes. Pour explorer ce phénomène :

1. Définir une procédure retourne(n) renvoyant un entier n "écrit à l'envers". Attention, elle devra fonctionner quel que soit le nombre de chiffres de n ! En déduire successivement :
2. une procédure suivant(n) effectuant *une* étape du processus décrit ci-dessus ;
3. une procédure estPalindrome(n) retournant *true* si l'entier n est un palindrome et *false* sinon ;

4. une procédure nbEtapes(n) calculant le nombre d'étapes nécessaires, en partant de n, pour arriver à un palindrome.

Exercice 8 *Tours de Hanoï* ([1])

n disques percés d'un trou en leur centre son enfilés sur une colonne, du plus grand (vers le bas) au plus petit (en haut). Le jeu consiste à les déplacer sur une deuxième colonne, un par un, en utilisant une troisième colonne auxiliaire mais sans jamais poser un disque sur un autre *plus petit*. On symbolisera le déplacement d'un disque de la $i^{\text{ème}}$ à la $j^{\text{ème}}$ colonne par l'instruction :

print(`déplacement de `, i ,` à `, j) ;

(*attention ! guillemets à gauche — "leftquotes"*).

1. Si les colonnes sont numérotées de 1 à 3 et si $i,\ j \in [\![1,3]\!]$, $i \neq j$, quel est en fonction de i et j l'indice de la troisième et dernière colonne ? On pourra écrire une fonction autreQue := (i,j) -> (cet autre indice).

2. Ecrire une procédure hanoi(n,i,j) "déplaçant n disques de la colonne i à la colonne j".
(Penser à la récursivité.)

3. Combien cette procédure utilise-t-elle de mouvements pour déplacer n disques d'une colonne à une autre ? Peut-on faire calculer ce résultat par *Maple*® (voir l'aide sur rsolve) ? Est-il optimal ?

Exercice 9 *suite "GEB"* ([2])

On définit (?) une suite (u_n) par $u_1 = u_2 = 1$ et de proche en proche pour $n \geq 2$:

$$u_n = u_{n-u_{n-1}} + u_{n-u_{n-2}}$$

1. Écrire une procédure (récursive) GEB(n) effectuant le calcul du terme d'indice n de cette suite.

2. Écrire une procédure prenant comme argument l'entier n et effectuant le tracé du "nuage de points" (k, u_k) pour $1 \leq k \leq n$, afin d'essayer de préciser le comportement de la suite (u_n).

Exercice 10 *suite autodescriptive ou "look-and-say"*

Il s'agit de la suite (u_n) dont les premiers termes sont

$$1, 11, 21, 1211, 111221, ...$$

1. Quelle est la définition de cette suite ?

2. Écrire une procédure suivant prenant comme argument *l'un des termes u_n de la suite* et calculant u_{n+1}.
En déduire une procédure suite effectuant le calcul de u_n en fonction de n.

3. *Étude numérique.* Tester les propriétés suivantes :

   - (u_n) n'est pas stationnaire ;
   - La taille des termes augmente indéfiniment (la suite tend vers $+\infty$) ;
   - Le *taux d'accroissement* d'un terme au suivant $(\frac{u_{n+1}}{u_n})$ tend vers une limite finie non nulle α (en déterminer une valeur approchée).
   - On montre[3] que α est racine du polynôme[4] :

$$\begin{aligned} P(X) \;=\; & X^{71} - X^{69} - 2X^{68} - X^{67} + 2X^{66} + 2X^{65} \\ & + X^{64} - X^{63} - X^{62} - X^{61} - X^{60} - X^{59} \\ & + 2X^{58} + 5X^{57} + 3X^{56} - 2X^{55} - 10X^{54} \\ & - 3X^{53} - 2X^{52} + 6X^{51} + 6X^{50} + X^{49} \\ & + 9X^{48} - 3X^{47} - 7X^{46} - 8X^{45} - 8X^{44} \\ & + 10X^{43} + 6X^{42} + 8X^{41} - 5X^{40} - 12X^{39} \\ & + 7X^{38} - 7X^{37} + 7X^{36} + X^{35} - 3X^{34} \\ & + 10X^{33} + X^{32} - 6X^{31} - 2X^{30} - 10X^{29} \\ & - 3X^{28} + 2X^{27} + 9X^{26} - 3X^{25} + 14X^{24} \\ & - 8X^{23} - 7X^{21} + 9X^{20} + 3X^{19} - 4X^{18} \\ & - 10X^{17} - 7X^{16} + 12X^{15} + 7X^{14} + 2X^{13} \\ & - 12X^{12} - 4X^{11} - 2X^{10} + 5X^9 + X^7 \\ & - 7X^6 + 7X^5 - 4X^4 + 12X^3 - 6X^2 \\ & + 3X - 6 \end{aligned}$$

     Confirmer à l'aide de *Maple*® votre valeur approchée de α (sans vous tromper de racine !).

4. Envisager d'autres valeurs initiales de u_0. Quelle remarque peut-on faire ?

Exercice 11 *Nombres parfaits, amicaux.*

On donne la fonction divisors du package numtheory retournant sous la forme d'un ensemble les diviseurs (> 0) de l'entier n. Ainsi, divisors(9) donne $\{1, 3, 9\}$.

1. Écrire une fonction sigma(n) calculant la somme $\sigma(n)$ des diviseurs *stricts* de n (donc : $\sigma(9) = 4$).

Un nombre n est dit *parfait* si $\sigma(n) = n$. Par exemple, 6 est parfait : $\sigma(6) = 1 + 2 + 3 = 6$.

2. Écrire une fonction estParfait(n) retournant *true* ou *false* selon que l'entier n est parfait ou non. En déduire une construction (grâce à select) de la liste des entiers parfaits entre 1 et 1000. Que remarque-t-on ?

Deux nombres (distincts) m et n sont dits *amicaux* si $\sigma(m) = n$ et $\sigma(n) = m$.

3. Que peut-on dire dans ce cas de $\sigma(\sigma(m))$ (*resp.* $\sigma(\sigma(n))$) ? Utiliser cette remarque pour écrire deux procédures ayant pour but de déterminer

   (a) le "plus petit" couple de nombres amicaux ;

   (b) tous les couples de nombres amicaux dont l'un des termes est $\leq N$, N entier fixé.

[1] Rien à voir avec la capitale vietnamienne : ce problème a été inventé en 1882 par le mathématicien français Édouard Lucas.

[2] Introduite dans *Gödel, Escher, Bach : les Brins d'une Guirlande Eternelle*, D. R. Hofstadter, Interéditions 1985.

[3] John H. Conway, 1987

[4] Fichier `grospoly.mws`