TD1 – Récursivité

Exercice 1 *: Quelques fonctions de base*

- *Écrire une fonction récursive qui calcule la factorielle d'un nombre entier positif.*
- *En remarquant que* $n^2 = (n-1)^2 + 2n - 1$ *écrire une fonction récursive qui calcule le carré d'un nombre entier positif.*
- *Écrire une fonction récursive qui calcule le pgcd de deux nombres entiers positifs. On rappelle que le pgcd de deux nombres A et B est le même que le pgcd de b et de a mod b si b n'est pas nul.*
- *Écrire deux fonctions qui calculent de manière récursive la somme et le produit de deux nombres entiers A et B, positifs ou nuls. Pour calculer la somme, on dispose des relations suivantes :*

  $A + 0 = A$

  $A + B = (A + (B-1)) + 1$ *pour* $A,B \geq 0$

  Pour calculer le produit, on a les relations suivantes :

  $A * 0 = 0$

  $A * B = (A * (B-1)) + A$ *pour* $A, B \geq 0$

Exercice 2 *: Palindrome*

Ecrire une action récursive qui reconnaît si les éléments d'une chaîne de caractères forment un palindrome, c'est-à-dire un mot symétrique (sans tenir compte des espaces, accents, ponctuations, ou majuscules). Exemples de palindromes : "élu par cette crapule", "éric, notre valet, alla te laver ton ciré".

Exercice 3 *: Quick sort*

On souhaite programmer l'algorithme de tri rapide. Cet algorithme consiste dans un premier temps à partager le tableau en deux autour d'un pivot, afin qu'à la gauche du pivot il n'y ait que des éléments plus petits ou égaux au pivot et à droite que des éléments plus grands.

Le principe consiste à choisir un pivot (généralement le premier du tableau) à créer deux indices un égal au pivot (bas) et l'autre sur le dernier élément du tableau (haut). Les éléments haut et bas sont comparés et on échange les valeurs

dans le tableau si Tab[haut]<Tab[bas]. Lors de l'échange le pivot doit également changer de place on peut utiliser l'équation pivot=bas+haut-pivot.
Ensuite il faut soit monter la valeur bas si le pivot est en haut et descendre haut si le pivot est en bas. L'algorithme s'arrête lorsque bas et haut se croisent.
Le tableau est maintenant partitionné, il faut recommencer le tri sur la partie basse et sur la partie haute.

- *Implémenter cet algorithme.*

Exercice 4 : Notation polonaise

La notation polonaise, est une notation mathématique d'expressions arithmétiques où les opérandes (nombres) sont écrits après les opérateurs (+,-,,/). Cette notation est adaptée à un usage technique en informatique/électronique, et a pour caractéristique d'éviter l'utilisation de parenthèses.*
*Exemples : (1) 2*3 + 4 s'écrit + * 2 3 4 ; (2) 2 * (3 + 4) s'écrit * 2 + 3 4*

*Aussi pour évaluer un opérateur, il faut trouver deux opérandes après. Ainsi dans l'exemple 1 l'addition ne peut pas s'effectuer directement elle attend deux opérandes. Donc on évalue * avec 2 et 3 qui retournera un premier opérande et à ce moment + pourra s'effectuer avec le retour de * 2 3 et la variable 4. De même pour le second exemple la multiplication ne peut récupérer qu'un opérande et doit attendre le résultat de + pour réaliser son travail.*

- *A partir d'une chaine de caractère qui contient une expression en notation polonaise, créer la liste des opérandes et des opérateurs.*
- *Écrire une fonction qui teste si un élément de la liste est un opérateur.*
- *Proposez maintenant une fonction récursive qui évalue une expression en notation polonaise. La fonction recevra la liste correspondant à l'opération à traiter, elle retournera la valeur se trouvant en début de liste si celle-ci n'est pas un opérateur. Dans le cas contraire, elle devra récupérer l'instruction du début de la liste via, puis évaluer les deux opérandes utiles à son évaluation par deux appels récursifs à la fonction.*
- *Proposez maintenant une évaluation pour des expressions logiques en notation polonaise, opérateurs (and, or et not).*

Exercice 5 : *Elément majoritaire*

Soit T un tableau de n éléments. La seule opération que l'on peut effectuer consiste à vérifier si deux éléments sont égaux ou non. Un élément x est majoritaire si son cardinale (occurrence) est supérieure ou égale à n/2.

- *Écrire une fonction qui calcul le cardinal d'un élément x dans la liste L.*
- *Proposer un algorithme permettant de vérifier si T possède un élément majoritaire.*

Amélioration :

- *Si on a déjà compté le cardinal d'un élément x, il n'est plus nécessaire de le recalculer. Pour cela il faut créer une liste d'exclusion permettant de ne pas recalculer le cardinal d'un élément déjà calculé.*
- *De plus si l'on trouve un nouvel élément en position D, cela veut dire qu'il n'était pas présent entre 0 et D-1. Il est donc inutile de compter ses occurrences entre 0 et D-1, il suffit de partir de D.*
- *Enfin si un élément n'a pas été trouvé dans la première partie du tableau il ne peut être majoritaire.*

Algorithme récursif : On souhaite écrire une version récursive de cet algorithme. Pour cela on peut découper le tableau en deux sous tableau. Calculer récursivement quel est l'élément majoritaire dans le premier tableau. S'il en existe un on doit calculer son cardinale dans la seconde partie et vérifier qu'il est bien majoritaire sur tout le tableau. Si ce n'est pas le cas il faut faire la même démarche sur la seconde partie du tableau.

- *Donner la forme de fonction récursive.*
- *Proposer une version récursive de cet algorithme. S'il n'y a pas d'élément majoritaire la fonction retournera les valeurs -1, 0 alors que dans le cas contraire elle retourne l'élément majoritaire et son cardinal.*