TD 5 : RÉCURSIVITÉ

Exercice 1 (*Fonction puissance*)

1. Ecrire une fonction itérative `powerIt (a,b : entier)` qui retourne a^b.
2. Donner une définition récursive puis écrire la fonction récursive `powerRec ( a , b : entier )` qui retourne a^b.
3. Dessiner l'arbre de récursivité de `powerRec ( 3 , 4 )` .

Exercice 2 (*Coefficient binomial*)

Pour deux entiers naturels n et p tels que $0 \leq p \leq n$, les coefficients binomiaux donnent le nombre de sous-ensembles possibles de p élements d'un ensemble à n éléments.

Ils se notent $\binom{n}{k}$.

On a les trois propriétés suivantes :

$$\forall p \in N, \forall n \in N \quad \binom{n}{0} = 1 \text{ et } \binom{n}{n} = 1 \text{ et } \binom{n}{p} = \binom{n-1}{p-1} + \binom{n-1}{p}$$

En utilisant ces formules, écrire une fonction calculant le coefficient binomial de deux entiers sans utiliser de calcul de factoriel. Quel type de fonction doit-on utiliser ?

Exercice 3 (*Fonction récursive pour tableau*)

Les fonctions suivantes doivent être testées dans un programme principal en utilisant un tableau rempli aléatoirement.

1. Ecrire une fonction recursive permettant d'afficher un tableau passé en paramètre.
2. Ecrire une fonction récursive permettant de trouver le maximum d'un tableau passé en paramètre.
3. Ecrire une fonction recursive permettant de compter le nombre d'éléments divisibles par 3 dans un tableau :
   — En testant les valeurs du tableau en partant de la fin
   — En testant les valeurs du tableau en partant du début

Exercice 4 (*Somme des entiers*)

1. Écrire une fonction itérative `sommeEntiers(int a, int b)` qui retourne la somme de tous les nombres entiers se trouvant entre a et b inclus (exemple : $a = 7$ et $b = 10$ retourne $7 + 8 + 9 + 10 = 34$).
2. En effectuant la somme dans l'ordre croissant (c'est-à-dire en commençant à sommer les nombres les plus petits), donner une définition récursive de cette somme puis écrire la fonction `sommeEntiersCroissant(int a, int b)` associée qui calcule cette somme (on pourra choisir entre récursivité terminale ou non-terminale.).
3. En parcourant la somme dans le sens décroissant (c'est-à-dire en commençant à sommer les nombres les plus grands), donner une définition récursive de cette somme puis écrire la fonction récursive `sommeEntiersDecroissant(int a, int b)` qui calcule cette somme.
4. Proposer un programme principal pour tester les fonctions construites ci-dessus en prenant des valeurs de paramètres aléatoires entre 100 et 200 pour les paramètres de ces fonctions en ajoutant des conditions si nécessaire.

Exercice 5 (*Somme des chiffres*)

1. Ecrire une fonction itérative qui retourne la somme des chiffres d'un entier composé de plusieurs chiffres.
   Exemple : `somme3Chiffres(219)` => 12
2. Modifier la fonction précédente pour qu'elle fonctionne pour n'importe quel nombre entier.
   Exemple : `sommeChiffres(3807)` => 18
3. Donner la définition récursive et écrire une version récursive non terminale de la fonction.
4. Donner l'arbre de récursivité de votre fonction récursive non terminale avec en argument le nombre 6053.
5. Recommencer les 2 points précédents en utilisant une version récursive terminale de la fonction.