Devoir 1

Devoir à rendre le mercredi 31 mars 2010

Exercice 1 : programme mystère

Voici deux fonctions en pseudo-code :

1	Fonction mystere1(entier n) :	1	Fonction mystere2(entier n) :
2	Si n = 0 :	2	Si n = 0 :
3	Afficher(0)	3	Afficher(0)
4	Sinon :	4	Sinon :
5	Afficher(n)	5	mystere2(n - 1)
6	mystere1(n - 1)	6	Afficher(n)
7	Fin Si	7	Fin Si
8	Fin Fonction	8	Fin Fonction

Donner l'affichage de l'exécution de `mystere1(5)` et `mystere2(5)`. Expliquer brièvement les résultats obtenus.

Exercice 2 : recherche d'un élément dans un tableau

1. Donner un algorithme qui donne la position de la première occurrence d'un élément dans un tableau quelconque d'entiers. Déterminer la complexité en temps de l'algorithme. Donner l'exécution du programme appliqué au tableau

1	3	5	5	4	5	1	8	11	3
---	---	---	---	---	---	---	---	----	---

pour la recherche des éléments suivants : 1, 5 et 11.

2. Donner un algorithme qui s'exécute en temps logarithmique ($\Theta(\log(n))$) qui donne la position de la première occurrence d'un élément dans un tableau *trié* d'entiers. Justifier la complexité en temps de l'algorithme donné. Donner l'exécution du programme appliqué au tableau

1	1	3	3	4	5	5	5	8	11
---	---	---	---	---	---	---	---	---	----

pour la recherche des éléments suivants : 1, 5 et 11.

Exercice 3 : compositions d'entier

Une composition d'un entier n est une *liste* d'entiers strictement positifs telle que la somme de ses éléments est égale à n . Par exemple, les compositions de 4 sont

(4)
(3, 1)
(1, 3)
(2, 2)
(2, 1, 1)
(1, 2, 1)
(1, 1, 2)
(1, 1, 1, 1)

1. Donner toutes les compositions des entiers 1, 2, 3, 4 et 5.
2. Classer les compositions de 5 en fonction de la valeur du premier élément de leur liste. Mettre en relation cette classification avec la question 1. Que remarquez-vous ?
3. En déduire une fonction récursive en Java qui prend en paramètre un entier n et qui affiche toutes les compositions de n .
4. Donner une fonction récursive en Java qui prend en paramètre un entier n et qui compte les compositions de n sans les créer.
5. (Optionnel) Il existe 2^{n-1} compositions de l'entier n . Justifier ce fait.

Exercice 4 : partitions d'entier

Une partition de l'entier n est une *liste ordonnée* (en ordre décroissant) d'entiers strictement positifs telle que la somme de ses éléments est égale à n . Par exemple, les partitions de 4 sont

(4)
(3, 1)
(2, 2)
(2, 1, 1)
(1, 1, 1, 1)

L'objectif de cet exercice est de programmer la fonction

`void P(n)`

(où P désigne "partitions") qui affiche toutes les partitions de n .

Pour générer les partitions d'un entier donné n , il faut s'intéresser à un problème plus général : générer les partitions de n dont les éléments de sa liste sont tous plus petits qu'une valeur donnée k . Par exemple, les partitions de $n = 4$ bornées par $k = 2$ sont

(2, 2)
(2, 1, 1)
(1, 1, 1, 1)

Il va donc falloir dans un premier temps écrire la fonction

`void PB(n, k)`

(où PB désigne "partitions bornées") qui affiche toutes les partitions de n bornées par k .

1. Donner l'affichage que devrait produire les instructions suivantes :

PB(5, 1)	PB(4, 1)	PB(3, 1)	PB(2, 1)	PB(1, 1)
PB(5, 2)	PB(4, 2)	PB(3, 2)	PB(2, 2)	
PB(5, 3)	PB(4, 3)	PB(3, 3)		
PB(5, 4)	PB(4, 4)			
PB(5, 5)				

2. Classer les partitions de $n = 5$ bornées par $k = 3$ en fonction de la valeur de leur premier élément. Mettre en relation cette classification avec la question 1. Que remarquez-vous ?
3. Programmer de façon récursive la fonction PB .
4. Donner un fonction récursive en Java qui compte les partitions bornées sans les créer.
5. Programmer la fonction P en utilisant la fonction PB .