TD 5 – Itération/Récursion

Concepts informatiques (CI2)

2011–2012

1 Exponentiation

$x^n = x.x^{n-1}$ (lorsque $n > 0$) et $x^0 = 1$. En utilisant cette propriété, écrire en Java :

1. une fonction qui calcule x^n (où x et n sont des entiers), par la méthode itérative (sans appel récursif) ;
2. une fonction qui calcule x^n (où x et n sont des entiers), en utilisant la récursion

2 Exponentiation rapide

L'algorithme dit d'*exponentiation rapide* permet de calculer la $n^{\text{ème}}$ puissance d'un nombre plus efficacement qu'en le multipliant n fois par lui-même. Il repose sur les deux faits suivants :

$$\begin{aligned} x^{2n} &= (x^n)^2 \\ x^{2n+1} &= x\,(x^n)^2 \end{aligned}$$

1. Écrire en Java une fonction récursive prenant deux entiers x et n en argument, et renvoyant en résultat x^n, en appliquant récursivement celle des deux égalités qui correspond.
2. Supposons que l'on calcule 2^{65} à l'aide de votre fonction. Donner la suite des appels récursifs effectués, avec leur imbrication.
3. Combien y a t-il d'appel à la fonction provoqués lors du calcul de 2^{16} ? 2^{32} ? 2^{64} ? et par la méthode précédente (exercice précédent) ?

3 Fibonacci

La suite de Fibonacci, dûe à Leonardo Pisano, Léonard de Pise dit Fibonacci, est définie de façon à calculer le nombre d'individus d'une population évoluant de la façon suivante :
- on suppose que les individus ne meurent jamais ;
- au début il n'y a que deux individus non pubères ;
- les individus deviennent pubères au bout de deux mois ;
- deux individus pubères engendrent chaque mois deux individus non pubères.

Combien d'individus existe t-il au bout de n mois ?
Trouver la récurrence, et écrire en Java la version récursive du calcul de $F(n)$.
Remarque : Cette suite est liée au nombre d'or $\frac{1+\sqrt{5}}{2}$...

4 Ackermann

On définit la fonction d'Ackermann (Wilhelm Ackermann pour les intimes) de la façon suivante :

$$A(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ A(m-1,1) & \text{si } m > 0 \text{ et } n = 0 \\ A(m-1,A(m,n-1)) & \text{si } m > 0 \text{ et } n > 0 \end{cases}$$

- Calculez $A(0,4)$, $A(1,2)$, $A(2,1)$ et $A(3,0)$ (si quelqu'un veut calculer $A(4,1)$ qu'il le fasse chez lui, pour $A(4,2)$ nous lui conseillons d'obtenir une potion d'immortalité) ;
- Écrivez un programme permettant de calculer la valeur de la fonction pour tous les n et m positifs ou nuls ;

5 Le mystère de la méthode mystérieuse...

On suppose qu'on a défini une classe pour les listes d'entiers. Les instances de cette classe répondent aux méthodes suivantes :
- `boolean estVide()` qui permet de savoir si la liste comporte ou non des éléments ;
- `int premier()` qui renvoie la valeur de la tête de la liste ;
- `Liste suivant()` qui renvoie la liste constituée de la liste sans son élément de tête ;
- `void ajouteEnTete(int a)` qui ajoute un élément en tête de liste et dont la valeur est a ;

On dispose également d'un constructeur par défaut qui permet de créer une liste vide. On considère les deux méthodes suivantes :

```
static Liste temp (Liste l1, Liste l2){
   if (l1.estVide()) return l2;
   else l2.ajouteEnTete(l1.premier());
   return temp(l1.suivant(),l2);
}

static Liste mystere (Liste l){
   return temp(l,  new Liste());
}
```

- que renvoie un appel à `mystere(new Liste())` ?
- que renvoie un appel à `mystere(l)`, si `l` est une liste à un élément de valeur 12 ?
- que renvoie un appel à `mystere(l)`, si `l` est une liste à deux éléments : le premier de valeur 17 et le second de valeur 28 ?
- que calcule la méthode `mystere` ?