Université de Nice Sophia-Antipolis
DEUG MIAS-MP 1ère année
Examen d'Informatique
Septembre 2001

AUCUN DOCUMENT N'EST AUTORISE. Soignez votre copie (code indenté, réponses concises...) et utilisez votre brouillon pour mettre en ordre vos idées. Vous êtes notés sur ce que vous écrivez, pas sur ce que vous auriez voulu écrire. Soyez clairs, simple et précis dans vos réponses, une réponse ambiguë est une réponse fausse.

Exercice 1 : *l'invasion des nénuphars.*

Un jardinier plante un nombre `a` de nénuphars dans son étang chaque nénuphar est capable d'engendrer un autre nénuphar chaque année, la population de nénuphars double donc chaque année.

La population des nénuphars au cours des années suit donc une suite u_n définie par :

$$\begin{cases} u_0 = a \\ u_{n+1} = 2u_n \end{cases}$$

1. Programmez une version itérative (utilisant une boucle `for`) du calcul de la population sous la forme d'une fonction `int PopulationIterative(int a, int n)` prenant en arguments la population initiale `a` et le nombre d'années écoulées `n`. Si `n < 0` la fonction renvoie 0 sinon elle renvoie la population u_n de l'année `n`. *(1 point)*

2. Programmez une version récursive (utilisant un appel à elle même) du calcul de la population sous la forme d'une fonction `int PopulationRecursive(int a, int n)` prenant en arguments la population initiale `a` et le nombre d'années écoulées. Si `n < 0` la fonction renvoie 0 sinon elle renvoie la population u_n de l'année `n`. (*1 point)*

3. L'étang a une superficie de 60 m^2 et un nénuphar fait 20 cm^2. Ecrivez un programme principal qui utilise l'une des deux fonctions précédentes et une boucle pour trouver combien d'années il faudra en partant de 3 nénuphars pour couvrir tout l'étang, et afficher le résultat. *(2 points)*
   NB: $1m^2 = 10\,000\ cm^2$

4. L'algorithme précédent n'est pas performant, montrez en effet que le nombre d'opérations effectuées n'est pas proportionnel à `n`. Donnez et expliquez sa complexité. *(1 point)*

5. Afin de ne pas perdre de temps à recalculer ce qui a déjà été calculé, écrivez une fonction `int NombreAnnees(int a, int max)` qui, pour un nombre initial de nénuphars `a` et un nombre maximal de nénuphar `max`, donne le nombre d'années à attendre pour que les nénuphars aient envahi l'étang. On ferra bien attention à ne pas réitérer l'erreur de l'exemple précèdent et le nouvel algorithme doit avoir une complexité en `O(n)`. *(3 points)*
   NB: il ne vous sera pas utile de faire appel aux fonctions précédentes.

Exercice 2 : *A la pour-suite de Fibonacci.*

Vous connaissez tous la suite de Fibonacci :

n	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	...
f_n	0	1	1	2	3	5	8	13	21	...

qui est définie par :

$$\begin{cases} f_0 = 0 \\ f_1 = 1 \\ f_n = f_{n-1} + f_{n-2} \end{cases}$$

Nous souhaitons créer une structure de donnée permettant de se déplacer le long de cette suite en gardant le rang courant (il faudra donc mémoriser deux nombres de Fibonacci `a` et `b` consécutifs pour pouvoir avancer et reculer ainsi que le rang `n` de l'un d'eux, par exemple $a = f_n$). *On n'utilisera donc pas de tableau ni de vecteur pour stocker tous les termes de la suite, ni de fonction pour calculer le terme de rang n !*

1. Ecrivez la classe `Fibonacci` avec:
   - ses trois variables privées `a`, `b` et `n`, *(0,5 point)*
   - un constructeur `Fibonacci()` *(0,5 point)*
   - un accesseur `int getNombre()` qui retourne le nombre de Fibonacci courant `a` *(0,5 point)*
   - un accesseur `int getRang()` qui retourne le rang courant `n` (ex: 6 lorsque `a` vaut 8) *(0,5 point)*
   - une méthode `void next()` qui avance d'un cran dans la suite, faisant ainsi évoluer les trois variables privées *(1 point)*
   - une méthode `void previous()` qui recule d'un cran dans la suite, faisant ainsi évoluer les trois variables privées (si cela est possible !!!) *(1 point)*
   - une méthode `public String toString()` qui renvoie l'information sur la position courante sous forme de chaîne de caractères, par exemple : "F(6)=8" *(1 point)*

2. Ajoutez un constructeur `Fibonacci(int n)` qui utilise la méthode next() pour initialiser les variables privées au rang `n` de la suite de Fibonacci *(1 point)*

3. Donnez un programme principal qui crée une instance de la classe au rang 10, l'affiche, avance d'un rang, l'affiche, recule de deux rangs et l'affiche. *(1 point)*

4. Construire dans une classe séparée `FibonacciFrame` l'interface de présentation d'un élément de la suite de Fibonacci. Le déplacement le long de la suite se fait grâce à deux boutons et l'affichage se fait avec un `Label`. La fenêtre a un fond blanc, un titre "Fibonacci" et apparaît à la position (20,30). Elle contient trois composants :
   - Un bouton affichant " - " qui fait reculer d'un cran en utilisant la méthode `next()` définie dans la classe `Fibonacci`.
   - Un bouton affichant " + " qui fait avancer d'un cran en utilisant la méthode `previous()` définie dans la classe `Fibonacci`.
   - Un Label qui affiche le nombre en utilisant la méthode `toString()` de la classe `Fibonacci`.

NB: Une instance de la classe Fibonacci sera créée avec le constructeur par défaut et immédiatement affichée au rang 0.

(5 points)

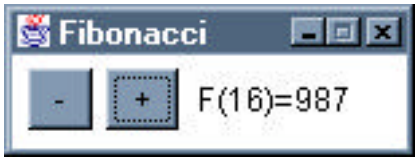

Figure 1. Copie d'écran de l'interface