Université Paris 7 - Denis Diderot
IF1 — Examen — 9 janvier 2007 — Durée : 3h
Documents, calculettes, ordinateurs et téléphones non autorisés
Les différents exercices sont indépendants

Exercice 1. On considère la séquence Java suivante :

```
boolean a, b, c, d, e;
    .......
if (a)
   d = b && c;
else
   d = true;
if (d)
   if(!b)
        e = (!a) || c;
   else
        e = true;
else
   e = a && b;
    .......
```

1.1. Quelles variables doivent être impérativement initialisées (dans la séquence d'instructions matérialisées par des points de suspension avant le premier if) pour que cette séquence n'engendre pas d'erreur de compilation ?

1.2. Déterminer, pour chacune des combinaisons des variables booléennes de la liste précédente, la valeur de la variable e après exécution de la séquence suivante et en déduire une séquence Java plus simple calculant la valeur de e.

Exercice 2. Soit le code Java suivant :

```
import fr.jussieu.script.*;
class Exam2 {
    public static void main(String[] args) {
      int v1 = 3, v2 = 6;
      if (v1 + 2 < v2) Deug.print("ric");
      if (v1 + 4 < v2) Deug.print("hoc");
      if (v1 + 4 > v2) Deug.print("het");
      Deug.println("ole");
    }
}
```

2.1. Quel doit être le nom du fichier contenant ce code source, quelle commande permet de le compiler, quel est le nom du fichier exécutable obtenu et comment réalise-t-on l'exécution ?

2.2. Déterminer ce qui est affiché à l'écran lorsqu'on exécute ce programme.

Exercice 3. Exécuter à la main la séquence suivante (dire en particulier ce qui est affiché à l'écran) :

```
int dest1 = 15, dest2 = 27;
int orig1 = 20, orig2 = 20;
boolean arrive1 = false;
boolean arrive2 = false;
while(!arrive1 && !arrive2) {
     orig1 = orig1 - 1;
     orig2 = orig2 + 2;
     arrive1 = (orig1 <= dest1);
     arrive2 = (orig2 >= dest2);
     Deug.println("Les voyageurs se trouvent en " + orig1 + " et " + orig2);
     }
Deug.println("Positions finales des voyageurs : " + orig1 + " et " + orig2);
```

Exercice 4. Ecrire un programme qui, après avoir lu un entier n tapé au clavier, lit n nombres réels (de type `float` correspondant à des intervalles de temps exprimés en secondes et fractions de secondes et calcule au fur et à mesure de leur lecture la hauteur dont un objet peut chuter dans le laps de temps correspondant.
On rappelle que l'accélération g de la pesanteur est égale à environ $9,81$ mètres par seconde au carré et que la distance parcourue dans un temps t est égale à $1/2 \times g \times t^2$.

Exemple d'exécution attendue :

```
Entrer le nombre de reels a lire: 3
Entrer un temps en seconde: 12.3
     hauteur de la chute: 742.0775 m
Entrer un temps en seconde: 6.25
     hauteur de la chute: 191.60158 m
Entrer un temps en seconde: 8
     hauteur de la chute: 313.92 m
```

Exercice 5. On considère la suite de nombres entiers définie par :

$$x_0 = 1,\ x_1 = 2,\ x_2 = 3 \text{ et } \forall i \geq 3, x_i = x_{i-1} + 2 \times x_{i-2} + 3 \times x_{i-3}.$$

5.1. Ecrire une fonction `element` qui, étant donné un entier n positif ou nul, calcule, sans utiliser de tableau, l'élément x_n de la suite et renvoie sa valeur.

5.2. Ecrire une fonction `tabElements` qui, étant donné un entier n positif ou nul, construit un tableau contenant les éléments x_0, x_1, ..., x_n et renvoie la référence de ce tableau. La fonction renverra `null` si le paramètre n est négatif.

Exercice 6. On souhaite écrire un programme qui permette de stocker des résultats sportifs—en l'occurrence, de volley-ball—dans un tableau récapitulatif. On utilisera pour cela un tableau `championnat` à deux dimensions d'entiers de type `int`.

Equipes	Victoires	Défaites	Sets Pour	Sets Contre	Points
France	2	1	6	6	5
Japon	0	3	4	9	1
Brésil	3	1	11	6	9

Rappelons qu'un match de volley-ball est composé de sets (ou « manches » en français) et se joue au meilleur des cinq sets, c'est-à-dire se termine dans l'une des situations suivantes :
- une équipe a gagné trois sets et l'autre un ou aucun : l'équipe gagnante marque 3 points et la perdante 0 point ;
- une équipe a gagné trois sets et l'autre deux : l'équipe gagnante marque 2 points et la perdante 1 point.

6.1. Ecrire une fonction `tableauInitial` qui prend en argument le nombre d'équipes et retourne un tableau championnat initial (avant qu'aucun résultat de match ne soit entré).

6.2. Ecrire une fonction `entrerMatch` qui prend en argument un tableau `championnat` et quatre entiers `n1` (numéro de l'équipe 1), `s1` (nombre de sets gagné par l'équipe 1), `n2` (numéro de l'équipe 2) et `s2` (nombre de sets gagnés par l'équipe 2) et qui met à jour le tableau en conséquence. La fonction renverra `-1` si une erreur est détectée (numéro d'équipe incorrecte ou score incorrect) et `0` sinon.

6.3. Ecrire une fonction `afficher` qui prend en argument un tableau `championnat` et affiche son contenu :

```
          Victoires  Defaites  Sets Pour  Sets Contre  Points
Equipe 0      2         1         6          6          5
Equipe 1      0         3         4          9          1
Equipe 2      3         1        11          6          9
```

6.4. Ecrire un programme `Championnat` qui permet de tester ces trois fonctions.

Exercice 7. Le but de cet exercice est de déterminer si, étant donné un tableau de nombres entiers positifs tous différents, un nombre donné peut s'écrire comme somme de deux d'entre eux.

7.1. Ecrire une fonction `tousDifferents` qui, étant donné un tableau t d'entiers, renvoie un booléen dont la valeur est `true` si tous les éléments du tableau sont différents et `false` si le tableau contient deux éléments identiques.

7.2. Ecrire une fonction `toutesLesPaires` qui, étant donné un tableau d'entiers t, renvoie `null` si les éléments de t ne sont pas tous différents et sinon un tableau de tableaux d'entiers contenant toutes les paires (non ordonnées) d'éléments de t.

Exemple : si $t =$ | 12 | 4 | 67 | 59 | , alors la fonction retourne le tableau de tableaux d'entiers suivant :

12	4
12	67
12	59
4	67
4	59
67	59

.

Afin d'allouer le tableau résultat, vous devez évaluer préalablement le nombre de paires possibles en fonction de la taille du tableau.
Si vous ne le pouvez pas, vous supposerez qu'il existe une fonction `static int nombrePaires(int[ ]t)` qui renvoie le nombre de paires qu'on peut former avec les éléments de t.

7.3. Ecrire une fonction `sommePaire` qui, étant donné un tableau d'entiers t tous différents et un entier x, renvoie

- une paire (a, b) d'éléments différents de t sous forme d'un tableau d'entiers telle que $x = a + b$ si tous les éléments du tableau sont différents et s'il existe une telle paire ;
- `null` si le tableau contient deux éléments identiques ou s'il n'existe pas de paire satisfaisante.

Exemple : si $t =$ | 12 | 4 | 67 | 59 | et $x = 71$, alors la fonction renvoie | 12 | 59 | (la paire | 4 | 67 | aurait également pu être renvoyée ; la fonction a retourné la première paire rencontrée qui convenait).
Si $t =$ | 12 | 4 | 67 | 59 | et $x = 10$, alors la fonction renvoie `null`.