Introduction à la Programmation (IP1)
Examen de seconde session – **Durée : 3 heures**

Université Paris-Diderot – Mercredi 22 juin 2016

- Aucun document ni aucune machine ne sont autorisés. Les téléphones doivent être éteints et rangés.
- Les exercices sont tous indépendants.
- Une réponse peut utiliser les fonctions ou les procédures attendues à une question précédente (même si elle est non traitée).
- Les fragments de code Java doivent être correctement indentés.
- On pourra écrire `t.length` pour accéder à la longueur d'un tableau.
- Il n'est pas nécessaire de vérifier les hypothèses faites sur les entrées des fonctions et procédures.
- Le barême est donné à titre indicatif. Il est donc sujet à de légères modifications.

Exercice 1 (Les amis Pythagoriciens – 4 points)

Un triplet d'entiers naturels strictement positifs (a, b, c) est appelé "triplet de Pythagore" si

$$a^2 + b^2 = c^2$$

Q1: *Écrire une fonction « `public static boolean isPythagorician (int a, int b, int c)` » qui attend trois entiers naturels strictement positifs `a`, `b` et `c` et qui retourne `true` si et seulement si le triplet (`a`, `b`, `c`) est un triplet de Pythagore. Par exemple, « `isPythagorician (3, 4, 5)` » retourne `true` tandis que « `isPythagorician (1, 2, 3)` » retourne `false`.*

Q2: *Écrire une fonction « `public static boolean havePythagoricianFriend (int a, int b, int m, int n)` » qui attend quatre entiers naturels `a`, `b`, `m` et `n` et qui retourne `true` s'il existe un entier `c` tel que $m \leq c \leq n$ et tel que le triplet (`a`, `b`, `c`) soit un triplet de Pythagore. Par exemple, « `havePythagoricianFriend (3, 4, 1, 10)` » retourne `true` et « `havePythagoricianFriend (3, 4, 1, 4)` » retourne `false`.*

Q3: *Écrire une fonction « `public static int nbPythagoricianTriples (int n, int m)` » qui attend deux entiers `n` et `m` et qui renvoie le nombre de triplets de Pythagore (`a`, `b`, `c`) tels que `a`, `b` et `c` sont compris entre `n` et `m`. Par exemple, « `nbPythagoricianTriples (1, 9)` » retourne 2 car les seuls triplets de Pythagore pour $(a, b, c) \in [1..9]^3$ sont $(3, 4, 5)$ et $(4, 3, 5)$.*

□

Exercice 2 (Tableaux de famille – 4 points)

Les n individus d'une famille sont représentés par les entiers naturels compris entre 0 et $n - 1$. Par exemple, la famille des 7 individus Paul, Pierre, Jeanne, Yoda, Luke, Leila, Darth est représentée par les entiers 0 pour Paul, 1 pour Pierre, 2 pour Jeanne, 3 pour Yoda, 4 pour Luke, 5 pour Leila et 6 pour Darth. La correspondance entre prénoms et numéros est représentée par un tableau `name` tel que `name[i]` contient le prénom de l'invididu de numéro `i`.

On représente aussi la relation de parentalité à l'aide de deux tableaux d'entiers. Le tableau `father` est tel que `father[i]` est le numéro de l'individu qui est le père de l'individu `i` ou `-1` si `i` n'a pas de père connu. Le tableau `mother` est tel que `mother[i]` est le numéro de l'individu qui est la mère de l'individu `i` ou `-1` si `i` n'a pas de mère connue. On pourra supposer qu'un individu ne peut pas être à la fois un père et une mère.

Q1: *Donnez le tableau `name` correspondant à la famille décrite plus haut.*

Q2: *Quels tableaux `mother` et `father` représentent exactement les informations suivantes ?*
Jeanne est la mère de Yoda et de Darth, Darth est le père de Luke et Leila, Leila est la mère de Pierre et Yoda est le père de Paul.

Q3: *Écrire une procédure « `public static void printRelation (int n, String[] name, int[] father, int [] mother, int i, int j)` » qui attend un entier naturel `n` représentant le nombre d'individus de la famille, un tableau de chaînes de caractères `name` organisé comme décrit plus haut, deux tableaux d'entiers `father` et `mother` organisés comme décrit plus haut, et deux numéros d'individus `i` et `j`. En supposant que l'individu `i` s'appelle `X` et que l'individu `j` s'appelle `Y`. Cette procédure affiche :*

— *"X est le père de Y"* si l'individu `i` est le père de l'individu `j`.

— *"Y est le père de X"* si l'individu `j` est le père de l'individu `i`.

— *"X est la mère de Y"* si l'individu `i` est la mère de l'individu `j`.

— *"Y est la mère de X"* si l'individu `j` est la mère de l'individu `i`.

— *"X et Y n'ont pas de relation directe"* si aucun des cas précédents n'est vérifié.

Pour répondre à cette question, vous pouvez utiliser `System.out.println` ou la procédure « `public static void printString (String m)` » qui affiche la chaîne `m` sur la sortie standard.

Q4: *Écrire une fonction « `public static int nbChildren (int n, int[] father, int[] mother, int i)` » qui attend un entier naturel `n` représentant le nombre d'individus de la famille, deux tableaux d'entiers `father` et `mother` organisés comme décrit plus haut, et un numéro d'individu `i`. Cette fonction retourne le nombre d'enfants de l'individu numéro `i`.*

Q5: *Écrire une fonction « `public static int[] children (int n, int[] father, int[] mother, int i)` » qui attend un entier naturel `n` représentant le nombre d'individus de la famille, deux tableaux d'entiers `father` et `mother` organisés comme décrit plus haut, et un numéro d'individu `i`. Cette fonction retourne un tableau contenant les enfants de l'individu `i`.*

Q6: *En utilisant les fonctions des deux questions précédentes, écrire une fonction « `public static boolean isGrandParent (int n, int[] father, int[] mother, int i)` » qui attend un entier naturel `n` représentant le nombre d'individus de la famille, deux tableaux d'entiers `father` et `mother` organisés comme décrit plus haut, et un numéro d'individu `i`. Cette fonction retourne `true` si l'individu `i` est un grand-père ou une grand-mère.*

□

Exercice 3 (Imagerie binaire – 4 points)

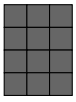
Dans cet exercice, on représente une image en noir et blanc par un tableau de booléens à deux dimensions. Un point noir est représenté par le booléen `true` et un point blanc par le booléen `false`.

Les images sont décrites ligne par ligne de la ligne la plus haute à la ligne la plus basse. Ainsi, la variable `img` suivante :

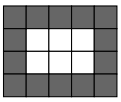
```
1  boolean[][] img = {
2    { false, false, true, false },
3    { false, false, false, true },
4    { false, true, true, true }
5  };
```

représente l'image de trois lignes et quatre colonnes suivante

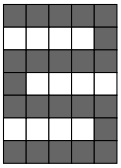
Q1: *Écrire une fonction « `public static boolean[][] allblack (int n, int m)` qui attend deux entiers positifs `n` et `m` et qui retourne l'image à `n` lignes et `m` colonnes constituées uniquement de points noirs. Par exemple, `allblack (4, 3)` produit la représentation de :*



Q2: *Écrire une fonction « `public static boolean[][] frame (int n, int m)` qui attend deux entiers positifs `n` et `m` et qui retourne l'image à `n` lignes et `m` colonnes constituées uniquement de points blancs exceptés sur les bords de l'image. Par exemple, `frame (4, 5)` produit la représentation de :*



Q3: *Écrire une fonction « `public static boolean[][] snake (int n, int m)` qui attend deux entiers positifs `n` et `m` et qui retourne l'image à `n` lignes et `m` colonnes dont les points noirs "serpentent" horizontalement en partant du coin en haut à gauche de l'image. Par exemple, `snake (7, 5)` produit la représentation de :*



□

Exercice 4 (Qui est-ce ? numérique – 8 points)

Dans cet exercice, on réalise un programme pour jouer au jeu "Qui est-ce ?" sur des nombres. Ce jeu se joue à deux. Chaque joueur a un nombre caché entre 0 et 99 que l'autre joueur doit découvrir. À tour de rôle, chaque joueur pose une question de la forme :

« Est-ce que ton nombre est divisible au moins k fois par n ? » [1]

En d'autres termes, le joueur choisit une valeur pour k et une valeur pour n qui lui permettent d'obtenir des informations sur le nombre de l'adversaire.

Chaque joueur a une grille de tous les nombres compris entre 0 et 99 sur laquelle il peut éliminer les nombres qu'il sait ne pas être le nombre de son adversaire. On représentera cette grille par un tableau de booléens de longueur 100. Si le booléen dans la case i vaut `true` alors i est peut-être le nombre de l'adversaire, sinon i n'est pas le nombre de l'adversaire. Initialement, le tableau ne contient bien sûr que des booléens valant `true`. Un joueur a gagné s'il ne reste plus qu'un seul nombre non éliminé de sa grille.

- **Q1:** *À l'aide d'une boucle `while`, écrire une fonction « `public static boolean kdivides (int h, int k, int n)` » qui attend un entier `h` compris entre 0 et 99 et deux entiers positifs `k` et `n`. Cette fonction retourne `true` si `h` est divisible au moins `k` fois par `n`.*
- **Q2:** *Écrire une procédure « `public static boolean[] makeGrid ()` » qui retourne une grille où aucun nombre n'a été éliminé.*
- **Q3:** *Écrire une fonction « `public static boolean hasWon (boolean[] grid)` » qui attend un tableau représentant la grille d'un joueur et qui retourne `true` si et seulement si le joueur a gagné.*
- **Q4:** *Écrire une procédure « `public static void eliminate (boolean[] grid, int k, int n, boolean a)` » qui attend un tableau représentant la grille d'un joueur, un entier k et un entier n et qui prend en compte la réponse `a` obtenue en posant la question « Est-ce que ton nombre est divisible au moins k fois par n ? » au joueur adverse. En d'autres termes, cette procédure élimine de `grid` tous les nombres incompatibles avec la réponse `a`.*
- **Q5:** *Écrire une procédure de jeu « `public static void play (int hA, int hB)` » qui attend le nombre secret `hA` d'un joueur A et le nombre secret `hB` d'un joueur B. Cette procédure repète un tour de jeu entre les deux joueurs A et B tant qu'aucun des deux joueurs n'a gagné. Un tour de jeu suit les étapes suivantes :*
  - *(a) On demande k au joueur A.*
  - *(b) On demande n au joueur A.*
  - *(c) On calcule la réponse à la question de divisibilité avec k et n comme paramètres.*
  - *(d) On prend en compte cette réponse dans la grille de A.*
  - *(e) On demande k au joueur B.*
  - *(f) On demande n au joueur B.*
  - *(g) On calcule la réponse à la question de divisibilité avec k et n comme paramètres.*
  - *(h) On prend en compte cette réponse dans la grille de B.*
  - *(i) Si les deux joueurs ont gagné, on arrête en affichant `"Egalité"`.*
  - *(j) Si l'un des deux joueurs a gagné, on affiche `"Le gagnant est X"` si `X` est le gagnant.*

  Vous devez introduire des procédures auxiliaires pour éviter de dupliquer du code similaire.

  Pour répondre à cette question, vous pouvez utiliser `System.out.println` ou la procédure « `public static void printString (String m)` » qui affiche la chaîne `m` sur la sortie standard. Vous utiliserez aussi la procédure « `public static int readInt ()` » qui retourne un entier écrit par l'utilisateur.
- **Q6:** *Écrire une fonction « `public static int[] bestChoice (boolean[] grid)` qui attend un tableau représentant la grille d'un joueur et qui retourne un tableau de taille 2 tel que la première case contient un entier k et la seconde case un entier n et tel que la question de divisibilité avec ces deux paramètres est la meilleure question que peut poser le joueur pour maximiser ses chances de gagner. Indice : une meilleure question élimine en moyenne le maximum de nombres quelque soit la réponse qu'on lui donne.*
- **Q7:** *Écrire une fonction « `public static int nbStepsToBeFound (int n)` » qui attend un nombre `n` entre 0 et 99 et qui retourne le nombre de tours de jeu nécessaires pour le trouver en utilisant la fonction `bestChoice` précédente.*

□

1. On rappelle que x est divisible par n si le reste de la division de x par n vaut 0. Quand x est divisible par n, on peut se demander si x est divisible 2 fois par n en testant si le reste de la division de $\frac{x}{n}$ par n vaut 0 et ainsi de suite...