Introduction à la Programmation (IP1)
Partiel – **Durée : 2 heures**

Université Paris-Diderot – Samedi 29 octobre 2016

- Aucun document ni aucune machine ne sont autorisés. Les téléphones doivent être rangés.
- Les exercices sont tous indépendants.
- **Attention** : Indiquez au début de votre copie quel langage de programmation vous utiliserez dans le restant de votre copie. Il n'est pas possible de changer de langage une fois celui-ci choisi. Pour rappel, les filières MATHS et MIASHS doivent composer en PYTHON tandis que les filières MATHS-INFOS et INFOS doivent composer en JAVA.
- Par convention, dans les énoncés, les codes sources JAVA se trouvent à gauche et les codes sources PYTHON à droite.
- Une réponse peut utiliser les réponses attendues à une question précédente (même si elle est non traitée).
- Les fragments de code doivent être correctement indentés.
- Dans les énoncés, on propose parfois une liste de fonctions et procédures utilisables. Vous pouvez cependant utiliser tout équivalent JAVA ou PYTHON. (Par exemple, `System.out.print` à la place de `printString`.) Attention cependant à ne pas utiliser des fonctions de la bibliothèque standard qui n'auraient pas été abordées en cours.

Exercice 1

1. *Écrire une fonction «* sum2 *» qui prend trois entiers x, y et z en paramètre et qui renvoie la somme des carrés de ces trois nombres. Par exemple, «* sum2 (3, 2, 1) *» s'évalue en* 14 *puisque* $3*3+2*2+1*1=14$.
2. *Écrire une procédure «* showSum3 *» qui prend trois entiers x, y et z en paramètre et affiche «* $x*x+y*y+z*z=s$ *» où x, y et z sont remplacés par leurs valeurs respectives et où s est remplacé par le résultat de «* sum2 (x, y, z) *». Par exemple, «* showSum2 (3, 2, 1) *» affiche :*

   `3 * 3 + 2 * 2 + 1 * 1 = 14`

   En JAVA*, vous pouvez utiliser par exemple :*
   - *«* `public static String intToString (int x)` *» qui renvoie une chaîne représentant l'entier* x.
   - *«* `public static void printString (String msg)` *» qui affiche la chaîne* msg.

□

Exercice 2

1. *Qu'écrire à la place de* A, B, C, D *et* E *pour que la fonction «* f *» suivante calcule le produit des nombres compris entre* a *et* b *? Par exemple, «* f (3, 5) *» renvoie* 60 *car* $3*4*5=60$.

```
public static A f (B a, C b) {
  int s = 1;
  for (int i = a; i D ; i++) {
    s = s * i;
  }
  E
}
```

```
def f (a, b) A
  s = 1
  for i B range (a, C , D ):
    s = s * i
  E
```

2. *Quelles sont les valeurs des variables* a, b *et* c *à la fin de l'exécution des instructions suivantes ?*

```
int x = 3;
int y = x * x;
int a = y / 2;
String c = "Dolores";
if (x < y || ! (x < y)) {
  c = c + " is ";
} else {
  c = c + " is not ";
}
c = c + "an AI.";
```

```
x = 3
y = x * x
a = y // 2
c = "Dolores"
if (x < y) or not (x < y):
  c = c + " is "
else:
  c = c + " is not "
c = c + "an AI."
```

□

Exercice 3 *En* JAVA, *pour les questions suivantes, vous pouvez utiliser par exemple :*

— *«* String characterAtPos (String s, int i) *» qui renvoie la chaîne de longueur* 1 *à la position* i *de* s.
— *«* int stringLength (String s) *» qui renvoie la longueur de la chaîne* s.
— *«* boolean stringEquals (String s1, String s2) *» qui renvoie* true *si les deux chaînes* s1 *et* s2 *sont égales.*

1. *Écrire une fonction «* count *» qui prend une chaîne* s *et une chaîne* a *de longueur* 1. *Cette fonction renvoie le nombre de fois où* a *apparaît dans* s.
2. *Écrire une fonction «* follows *» qui prend une chaîne* s *et deux chaînes* a *et* b *de longueur* 1 *en paramètre. Cette fonction renvoie le booléen vrai si et seulement si* a *est toujours immédiatement suivie de* b *dans* s. *Par exemple, «* follows ("Arnold Barney", "r", "n") *» renvoie vrai, et «* follows ("Arnold Barney", "n", "o") *» renvoie faux.*
3. *Écrire une fonction «* follows2 *» qui prend une chaîne* s *et trois chaînes* a, b *et* c *de longueur* 1 *en paramètre. Cette fonction renvoie le booléen vrai si* a *est toujours immmédiatement suivie de* b *dans* s *et si* b *est toujours immmédiatement suivie de* c *dans* s. *Par exemple, «* follows2 ("koali olili ", "a", "l", "i") *» renvoie vrai et «* follows2 ("ipoipa", "i", "p", "a") *» renvoie faux.*
4. *Écrire une fonction «* between *» qui prend une chaîne* s *et deux chaînes* a *et* b *de longueur* 1 *en paramètre. Cette fonction renvoie le booléen vrai si* a *est toujours immédiatement suivie et immédiatement précédé d'un* b *dans* s. *Par exemple, «* between ("koala alala", "l", "a") *» renvoie vrai et «* between ("plopl", "o", "l") *» renvoie faux.*

□

Exercice 4

1. *Écrire une fonction «* greaterThan *» qui prend une liste (en* PYTHON*) ou un tableau (en* JAVA*) d'entiers* a *ainsi qu'un entier* x *en paramètre et qui renvoie le booléan vrai si et seulement si tous les éléments de* a *sont strictement plus grands que* x. *Par exemple, «* greaterThan(t, 0) *» renvoie vrai si* t *est le tableau* { 1, 2, 3 } *en* JAVA *ou la liste* [1, 2, 3] *en* PYTHON *et renvoie faux si* t *est le tableau* { 1, 0, 3 } *en* JAVA *ou la liste* [1, 0, 3] *en* PYTHON.
2. *Écrire une fonction «* trio *» qui prend une liste (en* PYTHON*) ou un tableau (en* JAVA*) d'entiers* a *en paramètre. Cette fonction renvoie le booléen vrai si et seulement s'il existe une position* i *où trois éléments successifs de* a *sont tels que* a[i] $<$ a[i + 1] $<$ a[i + 2]. *Par exemple, «* trio (t) *» renvoie vrai si* t *est le tableau* { 1, 1, 2, 3 } *en* JAVA *ou la liste* [1, 1, 2, 3] *en* PYTHON *et renvoie faux si* t *est le tableau* { 1, 1, 0, 3 } *en* JAVA *ou la liste* [1, 1, 0, 3] *en* PYTHON.

□

Exercice 5

1. *Écrire une fonction* interleave *qui prend deux tableaux (en* JAVA*) ou listes (en* PYTHON*) d'entiers* a *et* b *en paramètre. On suppose que* a *et* b *sont de même longueur. Cette fonction renvoie un nouveau tableau (en* JAVA*) ou une nouvelle liste (en* PYTHON*) construit en insérant successivement un élément de* a *suivi d'un élément de* b. *Par exemple, si* a *est de longueur* 2 *et contient l'entier* 4 *suivi de l'entier* 9 *et que* b *contient l'entier* 7 *suivi de l'entier* 1 *alors cette fonction renvoie un tableau (en* JAVA*) ou une liste (en* PYTHON*) contenant* 4 *puis* 7 *puis* 9 *puis* 1.
2. *Écrire une fonction* intersection *qui prend deux tableaux (en* JAVA*) ou listes (en* PYTHON*) d'entiers* a *et* b *en paramètre et qui renvoie un nouveau tableau (en* JAVA*) ou une nouvelle liste (en* PYTHON*) de tous les entiers de* a *qui sont aussi dans* b.
3. *Écrire une fonction* noRepetition *qui prend un tableau (en* JAVA*) ou une liste (en* PYTHON*) d'entiers* a *en paramètre et qui renvoie un nouveau tableau (en* JAVA*) ou une nouvelle liste (en* PYTHON*) de tous les entiers de* a *mais sans répétition.*
4. *Écrire une fonction* intersectionNoRepetition *qui prend deux tableaux (en* JAVA*) ou listes (en* PYTHON*) d'entiers* a *et* b *en paramètre. On suppose que* a *et* b *sont* ***triés*** *(avec éventuellement des répétitions). Cette fonction renvoie un nouveau tableau (en* JAVA*) ou une nouvelle liste (en* PYTHON*) de tous les entiers de* a *qui sont aussi dans* b ***en supprimant les répétitions****. En profitant du fait que* a *et* b *sont triés, votre fonction devra être plus rapide que la composition de* intersection *et de* noRepetition.

□