université PARIS DIDEROT PARIS 7

EXAMEN CONCEPTS INFORMATIQUES CI2 16/05/2014 12^{00}-15^{00}

Aucun document ou support autre que le sujet ou les copies d'examen n'est autorisé
(la copie ou les brouillons du voisin ne sont pas des supports autorisés).
Positionner impérativement son mobile en mode « avion ».

1 Variables, Fonctions et Traductions

On considère les deux programmes `P` et `PR` suivants :

```
1  public class P {
2    public static int x;
3    public static int f(int a) {
4      int s = 0;
5      for (int i=0; i<=a; i++) s += i*i;
6      return s;
7    }
8    public static void main(String[] a) {
9      x = 6;
10     System.out.println("f("+x+")="+f(x));
11   }
12 }
```

```
1  public class PR {
2    public static int x;
3    public static int f(int a) {
4      if (a==0) return 0;
5      return a*a+f(a-1);
6    }
7    public static void main(String[] a) {
8      x = 6;
9      System.out.println("f("+x+")="+f(x));
10   }
11 }
```

1. Combien de fonctions sont-elles déclarées dans chacun des programmes `P` et `PR` ?
2. Combien d'appels de fonctions sont-ils présents dans le texte source des programmes `P` et `PR` ?
3. Combien d'appels de fonctions sont-ils réalisés à l'exécution des programmes `P` et `PR` ?
4. Quel est l'affichage produit à l'exécution des programmes `P` et `PR` ? Expliquer.

Si la valeur affectée à l'entier `x` était `10000` (ou une autre très grande valeur) et non `6` :

5. Cela poserait-il un problème à l'exécution de `P` ? Pourquoi ?
6. Cela poserait-il un problème à l'exécution de `PR` ? Pourquoi ?

Si l'unité de mesure de la mémoire d'une machine est le type concret `int` :

7. Combien de `int`, au minimum, sont-ils nécessaires pour assurer l'exécution du programme `P` ?
8. Combien de `int`, au minimum, sont-ils nécessaires pour assurer l'exécution du programme `PR` ?
9. Ces valeurs sont-elles dépendantes de la valeur de `x` ? Si oui, comment en dépendent-elles ?

10. La fonction `f` du programme `PR` est dite récursive. Pourquoi ?
11. La fonction `f` du programme `PR` n'est pas récursive terminale. Pourquoi ?
12. Transformer la fonction `f` du programme `PR` en une version récursive terminale.
13. Dérécursiver la fonction précédemment réécrite.

14. Traduire tel que vu en cours, le programme `P`.
15. Traduire tel que vu en cours, le programme `PR`.

2 Piles

Certains langages de programmation admettent `"while-"` et `"do-while"` comme schémas de boucle indéfinie. On dit qu'un tableau `e` de chaînes de caractères susceptible de contenir les mots `"do"` et `"while"` et les accolades `"{"` et `"}"` est *correct* au sens naturel où :

- à tout `"{"` correspond un unique `"}"` à sa droite et, inversement, à tout `"}"` correspond un unique `"{"` à sa gauche : on parle de *couple* ;
- à tout `"do"` correspond un unique `"while"` à sa droite (le premier disponible) : on parle de *couple*, mais certains `"while"` peuvent ainsi être *isolés* ;
- deux tels *couples* `"do"` ··· `"while"` et `"{"` ··· `"}"` ne peuvent s'intersecter :

  ni comme ça : `"do" "{" "while" "}"`,

  ni comme ça : `"{" "do" "}" "while"` ;

  ils peuvent par contre s'imbriquer.

Par exemple, `e1` = | `"while"` | `"do"` | `"while"` | et `e2` = | `"do"` | `"while"` | `"{"` | `"while"` | `"}"` | sont corrects, alors que `e3` = | `"do"` | `"while"` | `"do"` | et `e4` = | `"while"` | `"do"` | `"{"` | `"while"` | `"}"` | sont incorrects.

1. Parmi les tableaux suivants, expliquer lesquels sont corrects et lesquels sont incorrects.

   `f1` = | `"{"` | `"while"` | `"}"` | `"do"` | `"while"` |

   `f2` = | `"{"` | `"do"` | `"{"` | `"while"` | `"}"` | `"}"` |

   `f3` = | `"{"` | `"while"` | `"do"` | `"}"` | `"while"` |

   `f4` = | `"do"` | `"do"` | `"while"` | `"while"` | `"while"` |

   `f5` = | `"do"` | `"{"` | `"while"` | `"}"` | `"while"` |

2. Écrire une méthode `estCorrect` qui prend en argument un tableau `e` de chaînes de caractères, qui teste si `e` est correct et renvoie un booléen en conséquence.