université PARIS DIDEROT PARIS 7

EXAMEN CONCEPTS INFORMATIQUES CI2 20/06/2014 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).
Positionnez impérativement vos mobiles en mode « avion ».

1 Variables, Fonctions et Traductions

Soit le programme `Indien` suivant :

```
1  public class Indien {
2    public static int mystère(int n, int k) {
3      if (k==0) return 0;
4      if (k%2==0) return mystère(n*2,k/2);
5      return n+mystère(n*2,k/2);
6    }
7    public static void main(String []args) {
8      int x = 5, y = 10;
9      System.out.println("mystère("+x+","+y+")="+mystère(x,y));
10     System.out.println("mystère("+y+","+x+")="+mystère(y,x));
11   }
12 }
```

et soit le programme `Sioux` suivant :

```
1  public class Sioux {
2    public static int mystère(int n, int k) {
3      if (k==0) return 1;
4      if (k%2==0) return mystère(n*n,k/2);
5      return n*mystère(n*n,k/2);
6    }
7    public static void main(String []args) {
8      int x = 5, y = 3;
9      System.out.println("mystère("+x+","+y+")="+mystère(x,y));
10     System.out.println("mystère("+y+","+x+")="+mystère(y,x));
11   }
12 }
```

1. Quel est l'affichage produit par l'exécution du programme `Indien` ?
2. Quel est l'affichage produit par l'exécution du programme `Sioux` ?
3. Dessiner l'arbre des appels pour l'exécution du programme `Indien`, avec la valeur des paramètres et la valeur de retour de chaque appel.
4. Dessiner l'arbre des appels pour l'exécution du programme `Sioux`, avec la valeur des paramètres et la valeur de retour de chaque appel.
5. Que calculent les deux fonctions `mystère` ? Justifier rapidement.
6. Les deux fonctions `mystère` sont dites récursives, pourquoi ?
7. Pour chaque fonction `mystère` , indiquer si elle est récursive terminale ou non ? Justifier.
8. Pour $k \geq 0$, $n > 0$, combien d'appels sont-ils réalisés lors d'un appel initial à `mystère(`n`,`k`)` dans les deux cas ?

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

9. Combien de `int`, au minimum, sont-ils nécessaires pour assurer l'exécution du programme `Indien` ? Justifier.
10. Combien de `int`, au minimum, sont-ils nécessaires pour assurer l'exécution du programme `Sioux` ? Justifier.
11. Ces valeurs sont-elles dépendantes des valeurs de n et k ? Si oui, comment en dépendent-elles ? Justifier.
12. Transformer la fonction `mystère` du programme `Indien` en une version récursive terminale.
13. Dérécursiver la fonction précédemment réécrite.
14. Traduire tel que vu en cours, le programme `Indien`.

Balises

Un fichier XML est un fichier texte structuré comme ceci :

```
<concepts>
  <enseignants>
    <enseignant>Tartempion</enseignant>
    <enseignant>Machin</enseignant>
  </enseignants>
  <etudiants>
    <etudiant>Giulia</etudiant>
    <etudiant>Mohammed</etudiant>
    <etudiant>Patrick</etudiant>
  </etudiants>
</concepts>
```

Ce qui nous intéresse pour cet exercice sont les balises, c'est-à-dire les éléments structurants du fichier, comme `<etudiant>` ou `</enseignant>`. La particularité d'un document XML est que les balises viennent par couples et qu'à toute balise ouvrante <balise> correspond nécessairement une balise fermante </balise> plus loin dans le fichier. D'autre part, entre deux balises ouvrante/fermante, on peut placer n'importe quel autre couple ouvrant/fermant que l'on veut et ce autant de fois que l'on veut (par exemple les deux couples `etudiants`/`etudiant`) ; mais il est interdit d'entremêler deux couples. L'autre particularité est qu'un document XML est toujours constitué d'un couple de balises qui englobent tout le reste (le couple `<concepts></concepts>` dans l'exemple).
On vous donne la méthode `String getNextTag()` qui appelée itérativement vous renvoie dans l'ordre les balises XML d'un fichier (le reste du fichier est proprement ignoré par cette fonction). Lorsqu'il n'y a plus de balises à lire, la référence renvoyée est simplement `null`.

1. Donner trois exemples de fichiers mal formés du point de vue de la structure XML. Attention, les malformations doivent être de nature différente.
2. Écriver une méthode `estCorrect` qui teste si le fichier lu via des appels à `getNextTag` est un document XML correctement formé.