université PARIS DIDEROT PARIS 7

EXAMEN Session 2 — CI2 — 26/6/2015 12^{00}–14^{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).
Éteignez impérativement vos mobiles.

Lorsque des calculs sont nécessaires, il est impératif de les présenter sur la feuille d'examen. Il est aussi nécessaire de **justifier** ses réponses.

1 Exercice

On souhaite représenter en Java une structure de données stockant les éléments par lignes et colonnes, comme :

```
a
b  c  d  e
f  g  h  i
j  k  l  m  n  o
p  q  r  s  t  u
v  w  x  y
z
```

La machine utilise 2 octets pour représenter les caractères et 8 octets pour une référence.
On souhaite manipuler une telle structure de données à l'aide d'une variable de nom `tab` :

1. Quel serait le type de la variable `tab` ?
2. Quelle(s) instruction(s) permettent de définir `tab` et son contenu ?
3. Dessiner à la manière du cours et des TD, le contenu de la mémoire correspondant à la variable `tab` et les données qui lui correspondent. Précisez les tailles et les zones mémoires qui correspondent aux différents objets.
4. Écrire une fonction prenant en paramètre le tableau `tab` et qui affiche tous les éléments du tableau les uns à la suite des autres (sans retour à la ligne) en commençant par les éléments de la première ligne, puis ceux de la seconde, etc.

2 Exercice

Soit le programme suivant :

```
1  public class Multi {
2    public static int a(int x,int y) {
3      if (y==0) return x;
4      return 1+a(x,y-1);
5    }
6    public static int m(int x, int y) {
7      if (y==0) return 0;
8      return a(x,m(x,y-1));
9    }
10   public static void main(String []args) {
11     System.out.println(a(5,2));
12     System.out.println(a(2,5));
13     System.out.println("---");
14     System.out.println(m(7,3));
15     System.out.println(m(3,7));
16   }
17 }
```

1. Qu'affiche ce programme à son exécution ?
2. Que calcule la fonction `a` ?
3. Pour chaque appel à la fonction `a` dans la fonction `main`, combien d'appels à `a` sont exécutés au total ?
4. Que calcule la fonction `m` ?
5. Pour chaque appel à la fonction `m` dans la fonction `main`, combien d'appels à `m` et à `a` sont exécutés au total ?
6. `a` et `m` sont des fonctions définies récursivement, sont-elles récursives terminales ?
7. Transformer la fonction `m` en une récursive terminale.
8. Dérécursiver la fonction `m`.
9. Définir une fonction récursive Java permettant de calculer x^y en utilisant la fonction `m`.

3 Exercice

Soit le programme suivant :

```
public class Lang {
  public static void main(String []args) {
    String resultat = new String();
    Stack<String> s = new Stack<>();
    for (int i=0; i<args.length; i++) {
      switch (args[i]) {
      case "*":
        int n = Integer.parseInt(s.pop());
        for (int j=1; j<n; j++) { s.push(s.peek()); }
        for (int j=1; j<n; j++) { s.push(s.pop()+s.pop()); }
        break;
      case "+":
        s.push(s.pop()+s.pop()); break;
      case "-":
        s.pop(); break;
      default:
        s.push(args[i]); break;
      }
    }
    System.out.println(s.pop());
  }
}
```

On rappelle que l'opération `peek()` permet d'obtenir l'élément en sommet de pile sans le retirer de la pile et qu'il existe une opération `empty()` permettant de tester si une pile est vide ou non (la valeur renvoyée est du type `boolean`.

1. Que produit l'exécution `java Lang n o` ? Dessiner la suite des états de la pile `s`.
2. Que produit l'exécution `java Lang n o m - b + +` ? Dessiner la suite des états de la pile `s`.
3. Que produit l'exécution `java Lang n o m - b + + 2 *` ? Dessiner la suite des états de la pile `s`.
4. Que se passe-t-il à l'exécution de `java Lang d a b + + +` ?
5. Modifier le programme de sorte qu'aucune erreur de manipulation de pile ne provoque un arrêt brutal de l'exécution.

4 Exercice

Soit le programme Java suivant :

```
public class Prog2 {
  public static int i;
  public static int j;
  public static int k;
  public static int f(int v) {
    while (v!=0 && v%2==0) v = v/2;
    return v;
  }
  public static void main(String []a) {
    i = 1;
    while (i<6) {
      j = i*(i-1);
      k = f(j);
      System.out.println(k);
      i++;
    }
  }
}
```

1. Qu'affiche le programme si on l'exécute ?
2. De combien de variables ce programme a t-il besoin pour fonctionner ?
3. Traduire, tel qu'en cours et TDs, le programme en un autre programme ne contenant qu'un tableau d'entiers appelé `memoire`, un entier appelé `ins`, qu'une seule boucle `while`, une seule instruction `switch` (et pas de boucle `for` évidemment !!!) et éventuellement une pile d'entiers.