Université Paris Diderot – Paris 7
Concepts informatiques (CI2)
Examen du 18 mai 2010 - Durée : 2 heures 30 minutes
Documents, calculettes, ordinateurs et téléphones non autorisés.

Exercice 1. Quels sont les résultats affichés par l'exécution du code suivant (justifiez vos réponses) ?

```
class Param {
  static int fonc1(int a, int b) {
       a = a + 1; b = b + 2; return(a + b);}
  static void fonc2(int[] a, int[] b, int c) {
       int d = a[0]; a[0] = b[0]; b[0] = c; c = d;}
  static void fonc3(int[] a, int[] b, int[] c) {
       int d = a[0]; a[0] = b[0]; b[0] = c[0]; c[0] = d;}
  static void fonc4(int[][] a, int[] b) {
       int[] d = a[0]; a[0] = b; b = d; }
  public static void main(String[] arg) {
       int a = 1, b = 2, c = fonc1(a, b);
       System.out.println(a + " " + b + " " + c);
       int[] x = {1}, y = {2}; c = fonc1(x[0], y[0]);
       System.out.println(x[0] + " " + y[0] + " " + c);
       c = 3; fonc2(x, y, c);
       System.out.println(x[0] + " " + y[0] + " " + c);
       x[0] = 1; y[0] = 2; int[] z = {3};
       fonc3(x, y, z);
       System.out.println(x[0] + " " + y[0] + " " + z[0]);
       int[][] u = {{1}}; int[] v = {2}; fonc5(u,v);
       System.out.println(u[0][0] + " " + v[0]);
  }
}
```

Exercice 2. On dispose sur une assiette d'un tas de crêpes toutes de taille différente dont on souhaite faire une pile par taille décroissante (la plus grande au fond de la pile, c'est-à-dire directement sur l'assiette). Pour ce faire, on dispose d'une palette et la seule opération possible est de placer cette palette entre deux crêpes et de renverser la pile qui se trouve au-dessus de la palette.

On désignera chacune des n crêpes par un entier entre 0 (la plus petite) et $n-1$ (la plus grande). Ainsi, partant de la pile de 5 crêpes `[2 4 1 0 3[` (crêpe 2 au fond de la pile, crêpe 4 juste au-dessus, etc.), on souhaite obtenir sur l'assiette la pile `[4 3 2 1 0[`.
La crêpe au fond de la pile est dite de profondeur `0`, celle au-dessus de profondeur `1`, etc.
On désigne par

- `prof(c)` la profondeur de la crêpe dont le numéro est `c` ;
- `renverser(p)` l'opération consistant à renverser la pile de crêpes commençant à la profondeur `p` (c'est-à-dire les (p+1) crêpes du dessus).

Sur l'exemple, `prof(4)=3` et l'opération `renverser(3)` transforme la pile `[2 4 1 0 3[` en `[2 3 0 1 4]`.

`2.1.` Écrire la méthode `prof` sur une pile en n'utilisant que les opérations `push` et `pop`. On pourra utiliser une pile auxiliaire, mais, au retour de la fonction, la pile de crêpes retrouvera sont état initial.

`2.2.` Écrire la fonction `renverser` réalisant le renversement des crêpes sur une pile à partir d'une profondeur donnée (et ne changeant rien en dessous de cette profondeur) comme une suite d'opérations `pop`, `push` et `empty`. On pourra utiliser deux piles annexes.

`2.3.` Donner une suite de manipulations de type `renverser` permettant de transformer la pile `[2 4 1 0 3[` de l'exemple en la pile `[4 3 2 1 0[`.

`2.4.` Ecrire une fonction `crepes` écrite sous forme récursive qui, étant donnée une pile quelconque de n crêpes, affiche une suite de renversements plaçant les crêpes par taille décroissante sur cette pile.

Exercice 3.

3.1. Qu'affiche le programme suivant (justifiez votre réponse).

```
class Exo1 {
  static long f(int m, int n) {
      System.out.println("f(" + m + "," + n +  ")");
      if (m <= 0) return 1;
      if (m < n) return (1 + g(m, n-1));
      else return f(m-1,n) + g(m-1, n-1);
  }
  static long g(int m, int n){
      System.out.println("g(" + m + "," + n +  ")");
      if (m <= 0 || n <=0) return 0;
      if (m == n) return 1 +g(m-1, n);
      else return f(m-1, n-1) + g(m, n-1);
  }
  public static void main(String[] args){
    System.out.println(f(3, 5));
  }
}
```

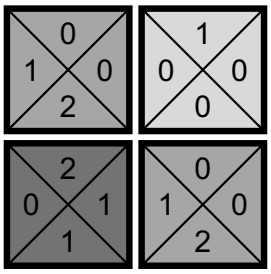
3.2. Adapter ce programme pour éviter qu'il ne refasse des calculs déjà faits.

Exercice 4. Les dominos de Wang sont des carrés tous de même taille avec un numéro sur chaque bord qui peuvent être agencés bord à bord sur une grille carrée de telle sorte que deux bords adjacents devront porter le même numéro. Les rotations et symétries de dominos sont interdites.

Le but de cet exercice est d'écrire un programme qui recherche par la méthode du backtracking toutes les façons de remplir une grille carrée de côté `N` avec des copies de dominos issus d'un ensemble `dom` fixé.

Un domino sera donné par un tableau de quatre entiers donnant les numéros portés par les bords Est, Nord, Ouest et Sud dans cet ordre. L'ensemble `dom` sera vu comme un tableau de tels tableaux.

Par exemple, avec `int[][] dom0 = {{0,1,0,0},{0,0,1,2},{1,2,0,1}};`, on pourra obtenir une grille de côté 2 comme celle-ci :



4.1. Expliquer pourquoi ce problème peut être résolu au moyen du backtracking et en particulier ce qu'est une solution partielle.

4.2. Développer la suite des essais que réalisera typiquement un algorithme de backtracking pour remplir une grille carrée de côté 3 avec l'ensemble `dom0` de l'exemple ci-dessus.

4.3. Écrire une méthode `possibles` qui prend en argument une solution partielle `t`, un entier `n` et un ensemble `dom` et qui retourne l'ensemble des dominos que l'on peut adjoindre à `t` pour obtenir une solution partielle plus longue.

4.4. Conclure en écrivant une méthode `backtrack` recherchant toutes les solutions pour un ensemble `dom` donné de dominos et un entier `N`.