UNIVERSITÉ PARIS 7 DENIS DIDEROT - LICENCE 1 - CI2 - EXAMEN - 20 JUIN 2011

Aucun document. Aucune machine.

Exercice 1 Expliquer ce que produit l'exécution du programme suivant :

```
class Vec {
  static int d1(int a, int b, int c, int d) {
    b = a + b;  d = c + d;
    return a * d - b * c;
  }
  static int d2(int[] t) {
    t[1] = t[0] + t[1]; t[3] = t[2] + t[3];
    return t[0] * t[3] - t[1] * t[2];
  }
  static int d3(int[][] t) {
    t[1][0] = t[0][0] + t[1][0];  t[1][1] = t[0][1] + t[1][1];
    return t[0][0] * t[1][1] - t[1][0] * t[0][1];
  }
  static int[] v1(int[][] t) {
    int[] v = new int[3];
    v[0] = d1(t[1][0],t[2][0],t[1][1],t[2][1]);
    v[1] = d1(t[2][0],t[0][0],t[2][1],t[0][1]);
    v[2] = d1(t[0][0],t[1][0],t[0][1],t[1][1]);
    return v;
  }
  static int[] v2(int[][] t) {
    int[] v = new int[3], a = new int[4];
    for (int i=0; i<3; i++){
      a[0] = t[(i+1)%3][0]; a[2] = t[(i+1)%3][1];
      a[1] = t[(i+2)%3][0]; a[3] = t[(i+2)%3][1];
      v[i] = d2(a);
      }
    return v;
  }
  static int[] v3(int[][] t) {
    int[] v = new int[3];
    int[][] a = new int[2][];
    for (int i=0; i<3; i++){
      a[0] = t[(i+1)%3];
      a[1] = t[(i+2)%3];
      v[i] = d3(a);
    }
    return v;
  }
  public static void main(String[] arg) {
    int[] x = {1,2,3,5};
    int e = d1(x[0], x[1], x[2], x[3]);
    System.out.println(x[0]+" "+x[1]+" "+x[2]+" "+x[3]+" | "+e);
    e = d2(x);
    System.out.println(x[0]+" "+x[1]+" "+x[2]+" "+x[3]+" | "+e);
    int[][] w = {{1,2},{3,5},{4,6}};
    e = d3(w);
    System.out.println(w[0][0]+" "+w[1][0]+" "+w[0][1]+" "+w[1][1]+" | "+e+"\n" );
    int[] ee = v1(w);
    for (int i=0; i<3; i++) System.out.println(w[i][0]+" "+w[i][1]+" | "+ee[i]);
    System.out.println();
    ee = v2(w);
    for (int i=0; i<3; i++) System.out.println(w[i][0]+" "+w[i][1]+" | "+ee[i]);
    System.out.println();
    ee = v3(w);
    for (int i=0; i<3; i++) System.out.println(w[i][0]+" "+w[i][1]+" | "+ee[i]);
    System.out.println();
  }
}
```

Exercice 2 Décrire le calcul de `s(-1,2)` avec :

```
static int s(int x, int y){
        if (x+y>0){
                if (x<y) return s(x+1,y)+1;
                return s(x,y-1)+1;
        }
        if (x+y<0){
                if (x<=y) return s(x,y+1)+1;
                return s(x-1,y)+1;
        }
        if (x > 1) return s(x-2,y+1)+1;
        if (x < 0) return s(x+1,y)+1;
        return x;
}
```

Exercice 3 On dispose d'un jeton et d'une grille carrée dont chaque case porte un entier positif (on utilisera une variable globale `grille` de type tableau de tableaux d'entiers). On déplace le jeton depuis une case du bord inférieur vers une case du bord supérieur, en respectant la règle suivante. À chaque étape, on peut placer le jeton sur l'une des cases suivantes :

- celle juste au-dessus,
- celle située une position plus haut et une position plus à gauche (à condition que le jeton ne soit pas déjà dans la colonne la plus à gauche),
- celle située une position plus haut et une position plus à droite (à condition que le jeton ne soit pas déjà dans la colonne la plus à droite).

Chaque fois que l'on passe par une case, on reçoit la somme indiquée par l'entier associé.

1. Écrire une méthode auxiliaire `gainMax` qui prend en argument un indice de ligne `i` et un indice de colonne `j` et qui renvoie le gain maximum quand on part de la case `(i,j)` de `grille`.
2. En déduire une méthode `gainMax` sans argument qui renvoie le gain maximum pour `grille`.
3. En dessinant l'arbre des appels, donner la trace de cette méthode pour la grille suivante :

1	2	3
6	5	4
7	8	9

4. Calculer le nombre d'appels récursifs dans le cas général.
5. Préciser comment améliorer ce programme en utilisant le concept de programmation dynamique.

Exercice 4 Pour le projet de deuxième année, `2n` étudiants doivent s'organiser en binômes. Les incompatibilités sont codées sous la forme d'un tableau `inc` de `2n` tableaux de `2n` booléens : la valeur de `inc[i][j]` indique si l'étudiant `i` a exprimé le souhait de ne pas être en binôme avec l'étudiant `j`. On souhaite rechercher par la méthode du backtracking une solution au problème pour un tableau d'incompatibilité donné.

1. Donner deux exemples raisonnables de tableaux `inc1` et `inc2`, l'un ne donnant aucune solution, l'autre au moins deux.
2. Choisir un moyen de coder les solutions (partielles et totales).
3. Écrire un programme répondant au problème.