

Puissance 3D

Prénom _____
 NOM _____

20

Interrogation de Java

(lundi 19 décembre 2005)

Remarques :

- ↪ vous travaillerez dans un répertoire `evaluations/i2`
- ↪ vous pouvez utiliser **toutes les notes** que vous voulez ainsi que les programmes qui se trouvent dans votre répertoire,
- ↪ l'interro se fait sur `linux1`,
- ↪ vous disposez de 2 heures,
- ↪ vous travaillez sur machine et remettez votre programme au terme du temps imparti.

L E PROBLÈME. Nous allons implémentez un jeu de *puissance 4* mais en 3 dimensions. Sur base d'un **cube** de taille 4, nous allons voir quand 3 pions sont alignés **verticalement** ou **horizontalement**.

Les règles du jeu sont simples, un joueur place un pion au sommet d'une colonne du cube (il doit donc choisir deux coordonnées pour ce faire ; la position "de face" et celle "en profondeur") et lache le pion qui tombe dans la colonne au dessus d'un autre pion éventuel.

Comme nous jouons en trois dimensions, le jeu se jouera à trois ¹, le joueur avec les pions **bleus**, celui avec les **rouges** et le dernier avec les **verts** .. que vous numéroterez de 1 à 3 car vous n'aurez pas le loisir d'afficher en couleurs.

1. Ecrire une fonction `entête` affichant un entête comme celui-ci (1 pt)

```
=====
P U I S S A N C E 3D
=====
```

2. Ecrire une fonction `main` contenant une déclaration d'un tableau de taille 4x4x4 d'entiers initialisés à 0. (2pts)

Déclarer une fonction `test(int[][][])` qui contiendra les tests que vous écrirez dans la suite et l'appeler à partir de la fonction `main`,

3. Ecrire une fonction `afficheCube(int[][][])` qui affiche le cube de (4pts)

¹Ne cherchez pas, il n'y a aucun lien de cause à effet

jeu .. sous la forme de 4 tableau à deux dimensions placés les uns à côté des autres,

```
0 0 0 0      0 0 0 0      0 0 0 0      0 0 0 0
0 0 0 0      0 0 0 0      0 0 0 0      0 0 0 0
0 0 0 0      0 0 0 0      0 0 0 0      0 0 0 0
0 0 0 0      0 0 0 0      0 0 0 0      0 0 0 0
```

Tester cette fonction en l'appelant depuis la fonction `test`,

4. Ecrire une fonction `placePion(int[][][], int, int, int)` qui place (4pts) un pion sur le cube.

Cette fonction reçoit en paramètre ; le *cube de jeu*, la *couleur du joueur*, la *colonne jouée* et la *profondeur*². Par exemple, les trois appels

```
↪ placePion(cube, 1, 2, 3),
↪ placePion(cube, 2, 2, 3) et
↪ placePion(cube, 3, 2, 0)
donneront la situation suivante,
```

```
0 0 0 0      0 0 0 0      0 0 0 0      0 0 0 0
0 0 0 0      0 0 0 0      0 0 0 0      0 0 0 0
0 0 0 0      0 0 0 0      0 0 0 0      0 0 2 0
0 0 3 0      0 0 0 0      0 0 0 0      0 0 1 0
```

5. Ecrire une fonction `aGagnant(int[][][])` recherchant un gagnant dans (6pts) le cube. Cette fonction retourne le numéro du gagnant s'il existe, 0 sinon.

Pour ce faire, la manière la plus simple (et sans doute la moins efficace) est de parcourir tout le cube. Pour chaque élément du cube, il faut tester si les deux éléments qui suivent en ligne, en colonne et en profondeur sont identiques au premier. Si c'est le cas, c'est que l'on a trois pions identiques alignés. La couleur en question est gagnante ... en résumé, trois boucles imbriquées contenant 3 tests.

6. Ecrire correctement les tests permettant de tester la fonction (3pts) `aGagnant(int[][][])`.
7. Prendre une copie de votre code chez vous et écrire la fonction `main` permettant de jouer. (0pt)

²Pour les distractifs, pas de notion de "ligne" puisque les pions tombent.