Nom, prénom, numéro d'étudiant : .....

Université Paris Sud, Licence MPI, Info 111 **Partiel du 3 novembre 2014 (deux heures)**

Calculatrices et autres gadgets électroniques interdits.
Seul document autorisé : une feuille au format A4 avec, au recto, la résumé de la syntaxe C++ de la fiche de TD 2 et, au verso, des notes manuscrites.
Les exercices sont indépendants les uns des autres ; il n'est pas nécessaire de les faire dans l'ordre.
Les réponses sont à donner sur le sujet. Les réponses aux questions à choix multiples doivent être données sur la dernière page. Des points négatifs sont affectés en cas de mauvaise réponse.

Exercice 1 (Cours)**.**

1. Rappeler la syntaxe et la sémantique de la boucle while en C++.

   Correction.
   Syntaxe :

```
   while (condition) {
bloc d instructions;
   }
```

   Sémantique : tant que la condition est vraie, on répète le bloc d'instructions :
   – La condition est évaluée
   – Si sa valeur est **true**, le bloc d'instructions est exécuté
   – On recommence

2. Quelles sont les étapes pour construire un tableau à deux dimensions en C++ ?

   Correction.
   Un tableau à deux dimensions se construit en quatre étapes :
   (a) Déclaration du tableau
   (b) Allocation du tableau
   (c) Allocation de chaque ligne
   (d) Initialization

   Illustrer votre réponse avec la construction d'un tableau à deux dimensions avec 7 lignes et 6 colonnes initialisé à zéro :

```
vector<vector<int>> grille (6);
for(int i=0; i<grille.size(); i++) {
    grille[i] = vector<int> (7);
    for(int j=0; j<grille[i].size(); j++)
        grille[i][j] = 0;
}
return grille;
```

Exercice 2 (Questions à choix multiples)**.**
Pour chaque (fragment de) programme suivant, sélectionner toutes les valeurs que peut entrer l'utilisateur pour que 42 soit affiché.

Question 1 ♣

```
vector<int> tab = {1, 1, 2, 5, 14, 42, 132};
int indice;
cin >> indice;
cout << tab[indice] << endl;
```

■ 5 B 3 C 4 D 1
E *Aucune de ces réponses n'est correcte.*

Question 2 ♣

```
int n;
cin >> n;
if ( n < 3 or n >= 5 ) {
    cout << 14 << endl;
} else {
    cout << 42 << endl;
}
```

■ 4 B 0 C 5 D 2 ■ 3
F *Aucune de ces réponses n'est correcte.*

Question 3 ♣

```
int s = 12;
int n;
int i = 0;
cin >> n;
while (i < n) {
    s = s+i*i;
    i++;
}
cout << s <<endl;
```

A 6 B 4 ■ 5 D 3
E *Aucune de ces réponses n'est correcte.*

Question 4 ♣

```
int mystere(int p) {
    if(p == 0) {
        return 1;
    }
    return 2*mystere(p-1);
}

int main() {
    int a;
    cin >> a;
    cout << mystere(a) + 10 << endl;
    return 0;
}
```

■ 5 B 3 C 16 D 32
E *Aucune de ces réponses n'est correcte.*

Question 5 ♣

```
    int a, b, c, d;
    a = 20;
    b = 20;
    cin >> c;
    cin >> d;
    int s = a;
    if(b != a) {
        s = s+b;
    }
    if (c != b and c != a) {
        s = s+c;
    }
    if (d != a and d != b and d !=c ) {
        s = s+d;
    }
    cout << s << endl;
```

■ 10 12 ■ 20 22 C 0 2 D 20 2 E 11 11
F *Aucune de ces réponses n'est correcte.*

Question 6 ♣

```
    vector<int> tab = {14, 3, 5, 6, 4, 3, 5, 0, 9};
    cin >> tab[7];
    int s = 0;
    for (int i=0; i<tab.size(); i++) {
        if (tab[i] % 3 != 0) {
            s = s + tab[i];
        }
    }
    cout << s << endl;
```

A 7 B 3 C −17 ■ 14
E *Aucune de ces réponses n'est correcte.*

Exercice 3 (Fonctions)**.**

1. Compléter le code de la fonction dont le prototype, la documentation et les tests sont donnés ci-dessous :

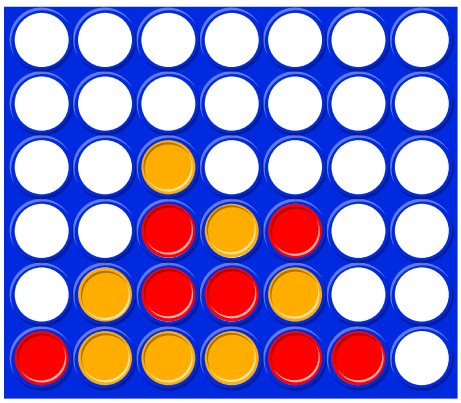
```
/** Somme des cubes
 *  @param n un entier positif
 *  @return 1^3+2^3+...+n^3
 **/
int sommeDesCubes(int n);

void sommeDesCubesTest() {
    ASSERT( sommeDesCubes(0) == 0 );
    ASSERT( sommeDesCubes(1) == 1 );
    ASSERT( sommeDesCubes(2) == 9 );
    ASSERT( sommeDesCubes(3) == 36 );
}

int sommeDesCubes(int n) {
    int result = 0;
    for (int i=0; i<=n; i++) {
        result = result + i*i*i;
    }
    return result;
}
```

2. ♣ Même chose, mais en utilisant une fonction récursive :

```
int sommeDesCubesRecursive(int n) {
    if (n==0) {
        return 0;
    } else {
        return n*n*n + sommeDesCubesRecursive(n-1);
    }
}
```

1 Problème

`Puissance 4` est un jeu où deux joueurs font tomber chacun à son tour un pion dans une grille verticale comptant 6 rangées et 7 colonnes, dans le but d'aligner 4 pions. Pour cet exercice, vous n'avez besoin ni de savoir jouer ni de connaître le détail des règles rappelées ci-dessous ; il suffit d'avoir compris le principe général.



« Chaque joueur dispose de 21 pions d'une couleur (par convention, en général jaune ou rouge). Tour à tour les deux joueurs placent un pion dans la colonne de leur choix, le pion coulisse alors jusqu'à la position la plus basse possible dans ladite colonne à la suite de quoi c'est à l'adversaire de jouer. Le vainqueur est le joueur qui réalise le

premier un alignement (horizontal, vertical ou diagonal) d'au moins quatre pions de sa couleur. Si, alors que toutes les cases de la grille de jeu sont remplies, aucun des deux joueurs n'a réalisé un tel alignement, la partie est déclarée nulle. »

Dans les exercices des pages suivante, on s'intéresse à l'écriture d'un programme pour jouer à Puissance 4. Par convention, on représentera par 1 le joueur 1 ou un de ses pions, par 2 le joueur 2 ou un de ses pions et par 0 une case vide.

Exercice 4 (Échauffement : puissance 4 sur une seule rangée)**.**

1. On commence par la fonction `mystere` suivante :

```
bool mystere(vector<int> zut, int hop) {
    int bla=0;
    for (int z=0; z<zut.size(); z++) {
        if (zut[z] == hop) {
            bla = bla + 1;
            if (bla == 4) {
                return true;
            }
        } else {
            bla = 0;
        }
    }
    return false;
}
```

Deviner ce que cette fonction est sensée faire, puis réécrire la fonction avec sa documentation, en choisissant des noms informatifs pour elle-même et pour ses variables.

```
/** Teste si 'tableau' contient 'valeur' dans au moins 4 cases consécutives
 *  @param tableau un tableau d'entiers à une dimension
 *  @param valeur  un entier
 *  @return un booléen
**/
bool estGagnante1D(vector<int> tableau, int valeur) {
    int compteur=0;
    for (int i=0; i<tableau.size(); i++) {
        if (tableau[i] == valeur) {
            compteur = compteur + 1;
            if (compteur == 4) {
                return true;
            }
        } else {
            compteur = 0;
        }
    }
    return false;
}
```

2. Compléter les tests pour la fonction `estGagnante1D` suivante, avec au moins deux autres appels pertinents à ASSERT. La fonction `mystere` est celle de la première question.

```
/** Renvoie le joueur gagnant
 *  @param  grille un tableau d'entiers 0, 1 ou 2
 *  @return un entier 1 ou 2 si la grille contient quatres pions
 *          consécutifs de cette couleur, et 0 sinon
**/
int gagnant1D(vector<int> grille) {
    for (int joueur=1; joueur<=2; joueur++) {
        if (mystere(grille, joueur)) {
            return joueur;
        }
    }
    return 0;
}

void gagnant1DTest() {
    ASSERT( gagnant1D({2,2,1,1,1,1,0,2}) == 1 );
    ASSERT( gagnant1D({2,2,2,2,1,1,0,1}) == 2 );
    ASSERT( gagnant1D({2,2,1,2,1,1,0,1}) == 0 );


}
```

Exercice 5.
Une grille de puissance 4 est représentée par un vector<vector<**int**> >, de sorte que grille [3] représente la quatrième rangée de la grille en partant du haut. Pour alléger les notations, on définit le raccourci suivant :

```
typedef vector<vector<int>> Grille;
```

1. Compléter le code de la fonction ligneGagnante suivante. On pourra, mais ce n'est pas obligatoire, réutiliser la fonction mystere de l'exercice précédent :

```
/** Teste si 4 pions de 'joueur' sont alignés sur une ligne
 * @param grille une grille de puissance 4
 * @param joueur un entier 1 ou 2 désignant le joueur
 * @return un booleen
 **/
bool ligneGagnanteCorrection(Grille grille, int joueur) {
    for(int i=0; i<grille.size(); i++) {
        int compteur = 0;
        for(int j=0; j<grille[i].size(); j++) {
            if (grille[i][j]==joueur) {
                compteur++;
                if (compteur==4)
                    return true;
            } else {
                compteur=0;
            }
        }
    }
    return false;
}
```

2. La partie principale du jeu est implantée par la fonction ci-dessous :

```
1  void puissance4Bogguee() {
2      Grille grille = grilleVide();
3      Grille grillePrecedente;
4      int colonne, x;
5      int joueur = 1;
6      do {
7          cout << "Joueur " << joueur
8               << ": entrer un numéro de colonne entre 1 et 7" << endl;
9          cin >> colonne;
10         grillePrecedente = grille;
11         grille = unCoup(grille, colonne, joueur);
12         if (grille == grillePrecedente) {
13             cout << "Coup invalide" << endl;
14         } else if (estGagnante(grille, joueur)) {
15             cout << "Félicitation joueur " << joueur
16                  << ", vous avez gagné!" << endl;
17             return;
18         }
19         joueur = joueur % 2 + 1;
20         affiche(grille);
21     } while (estPleine(grille));
22     cout << "Partie nulle" << endl;
23 }
```

Elle utilise, entre autres, la fonction unCoup documentée comme suit :

```
1  /** Joue un coup
2   * @param grille  une grille de puissance 4
3   * @param colonne un entier entre 0 et 6 désignant
4   * une colonne de la grille
5   * @param joueur  un entier 1 ou 2 désignant le joueur
6   * @return la grille après que 'joueur' ait mis un pion dans 'colonne'.
7   *         si le coup est invalide, la grille est inchangée
8   **/
9  Grille unCoup(Grille grille, int colonne, int joueur) {
```

La fonction puissance4Bogguee contient des bogues à corriger l'un après l'autre. Dans chaque cas, **identifier le numéro de la ligne fautive et comment la corriger**.

(a) Dès la fin du premier coup, la partie s'arrête.

Correction.

Ligne 21 :

```
    } while (not estPleine(grille));
```

(b) Quand le joueur met un pion dans la colonne 7, le coup est considéré comme invalide.

Correction.

Ligne 11 :

```
        grille = unCoup(grille, colonne-1, joueur);
```

(c) En cas de coup invalide, le programme ne redemande pas un coup au même joueur.

Correction.

Ligne 19 :

```
        } else {
            joueur = joueur % 2 + 1;
        }
```

Exercice 6 (♣).

1. Implanter la fonction unCoup documentée ci-dessus.

```
Grille unCoup(Grille grille, int colonne, int joueur) {
    if (grille[0][colonne] != 0 or colonne >= 7) {
        return grille;
    }
    int i=1;
    while ( i < 6 and grille[i][colonne] == 0 ) i++;
    grille[i-1][colonne] = joueur;
    return grille;
}
```

2. Implanter la fonction diagonaleDroiteGagnante documentée ci-dessous.

```
/** Teste si 4 pions de 'joueur' sont alignés sur une diagonale
 * droite :
 * 0 0 0 x
 * 0 0 x 0
 * 0 x 0 0
 * x 0 0 0
 *
 * @param grille une grille de puissance 4
 * @param joueur un entier 1 ou 2 désignant le joueur
 * @return un booleen
 **/
```

```
bool diagonaleDroiteGagnante(Grille grille, int joueur) {
    for(int i=3; i<grille.size(); i++) {
        for(int j=0; j<grille[i].size() - 3; j++) {
            int k;
            for(k=0; k<4 and grille[i-k][j+k] == joueur; k++);
            if (k==4)
                return true;
        }
    }
    return false;
}
```

Nom, prénom, numéro d'étudiant : . . . .

Feuille de réponses, Questions à Choix Multiples, 3/11/2014

Le QCM est corrigé automatiquement, ce qui impose quelques contraintes :

- **Les réponses sont à donner exclusivement sur cette feuille : les réponses données sur les feuilles précédentes ne seront pas prises en compte.**
- **Les cases doivent être entièrement grisées.**
- **Il est recommandé d'utiliser un crayon ou autre tout stylo effaçable.**
- **Effacer avec du blanc devrait marcher, à vos risques et périls.**

Coder votre numéro d'étudiant ici, un chiffre par ligne :

☐**0** ☐**1** ☐**2** ☐**3** ☐**4** ☐**5** ☐**6** ☐**7** ☐**8** ☐**9**

☐**0** ☐**1** ☐**2** ☐**3** ☐**4** ☐**5** ☐**6** ☐**7** ☐**8** ☐**9**

☐**0** ☐**1** ☐**2** ☐**3** ☐**4** ☐**5** ☐**6** ☐**7** ☐**8** ☐**9**

☐**0** ☐**1** ☐**2** ☐**3** ☐**4** ☐**5** ☐**6** ☐**7** ☐**8** ☐**9**

☐**0** ☐**1** ☐**2** ☐**3** ☐**4** ☐**5** ☐**6** ☐**7** ☐**8** ☐**9**

☐**0** ☐**1** ☐**2** ☐**3** ☐**4** ☐**5** ☐**6** ☐**7** ☐**8** ☐**9**

☐**0** ☐**1** ☐**2** ☐**3** ☐**4** ☐**5** ☐**6** ☐**7** ☐**8** ☐**9**

☐**0** ☐**1** ☐**2** ☐**3** ☐**4** ☐**5** ☐**6** ☐**7** ☐**8** ☐**9**

Question 1 : ■**A** ☐**B** ☐**C** ☐**D** ☐**E**

Question 2 : ■**A** ☐**B** ☐**C** ☐**D** ■**E** ☐**F**

Question 3 : ☐**A** ☐**B** ■**C** ☐**D** ☐**E**

Question 4 : ■**A** ☐**B** ☐**C** ☐**D** ☐**E**

Question 5 : ■**A** ■**B** ☐**C** ☐**D** ☐**E** ☐**F**

Question 6 : ☐**A** ☐**B** ☐**C** ■**D** ☐**E**

CORRECTION