Corrigé du partiel du 28 octobre 2013

Exercice 1. – Le fragment de programme suivant affiche : $0, 2$:

```
int x, y;
x = 5*(2/5);
y = (5*2)/5;
cout << x << ", " << y << endl;
```

– Le fragment de programme suivant affiche $10, 1$:

```
int x, y;
x = 0;
y = 0;
while ( x < 10 ) {
    x = x + 1;
    y = 1;
}
cout << x << ", " << y << endl;
```

– Le fragment de programme suivant affiche $3, 3$:

```
int x, y;
y = -3;
x = -y;
if ( y > 0 ) {
    y = 2*x;
} else {
    y = -y;
}
cout << x << ", " << y << endl;
```

– Le fragment de programme suivant affiche $0, 2$:

```
int f(int x, int y) {
    x = x+1;
    return y-1;
}

int main() {
    int x,y;
    x = 1;
    y = 2;
    x = f(y,x);
    cout << x << ", " << y << endl;
}
```

– Le fragment de programme suivant affiche $1, 2$:

```
void f(int x, int y) {
    x = x+1;
    y = y-1;
}

int main() {
    int x,y;
    x = 1;
```

```
    y = 2;
    f(y,x);
    cout << x << ", " << y << endl;
}
```

– Le fragment de programme suivant affiche $1, 2$:

```
int g(int x, int y) {
    x = x+1;
    return y-1;
}

int main() {
    int x,y;
    x = 1;
    y = 2;
    g(x,y);
    cout << x << ", " << y << endl;
}
```

Exercice 2 (Fonctions)**.**

(1) Implanter une fonction itérative `puissanceIterative(n, k)` qui reçoit deux entiers positifs en paramètres et renvoie n^k.

```
int puissanceIteratif(int n, int k) {
    int resultat = 1;
    for (int i=1; i<=k; i++)
        resultat = resultat * n;
    return resultat;
}

int puissanceIteratifTest() {
    ASSERT( puissanceIteratif(3, 0) == 1 );
    ASSERT( puissanceIteratif(3, 1) == 3 );
    ASSERT( puissanceIteratif(3, 2) == 9 );
    ASSERT( puissanceIteratif(3, 3) == 27 );
}
```

(2) Implanter une fonction récursive `puissanceRecursive(n, k)` ayant le même comportement.

```
int puissanceRecursif(int n, int k) {
    if (k==0)
        return 1;
    else
        return n * puissanceRecursif(n, k-1);
}
```

Exercice 3 (Jeu du pendu)**.** (1) Fonction `contient` :

```
/** Tester si un tableau contient une lettre donnée.
 * @param tab le tableau a tester
 * @param c la lettre
 * @return vrai si la lettre est dans le tableau.
 **/
bool contient(vector<char> mot, char c) {
```

```
    for (int i=0; i < mot.size(); i++)
        if ( mot[i] == c )
            return true;
    return false;
}
```

(2) Des tests pour masqueMot :

```
void masqueMotTest() {
    ASSERT( masqueMot("chat",    {'c'})       == "c***"    );
    ASSERT( masqueMot("bonjour", {'o', 'r'})  == "*o**o*r" );
    ASSERT( masqueMot("bonjour", {'h', 'k'})  == "*******" );
    ASSERT( masqueMot("livre",   {'l', 'i', 'v', 'r', 'e'}) == "livre" );
}
```

(3) Fonction `motTrouve` :

```
bool motTrouve(string mot, vector<char> lettres) {
    for (int i=0; i < mot.size(); i++)
        if ( not contient(lettres, mot[i]) )
            return false;
    return true;
}
```

(4) Les 3 erreurs dans :

```
1  bool partie(string mot){
2      vector<char> lettres = {};
3      int nb_essais = 0;
4      int nb_max = 7;
5      char lettre;
6
7      cout << "Bienvenu sur le Jeu du Pendu!" << endl;
8      do {
9          cout << "Mot mystère: "  << masqueMot(mot, lettres) << endl;
10         cout << "Il vous reste " << nb_max-nb_essais << " essais. "
11              << "Proposez une lettre: ";
12         cin >> lettre;
13
14         if ( mystere(lettres, lettre) )
15             cout << "Lettre déjà proposée ..." << endl;
16         else {
17             lettres.push_back(lettre);
18             if ( mystere(mot, lettre) )
19                 cout << "Bien vu!"<< endl;
20             else {
21                 nb_essais + 1;
22                 cout << "Raté, réfléchissez plus fort ..." << endl;
23             }
24             if ( not motTrouve(mot, lettres) )
25                 return true;
26         }
27     } while ( nb_essais <= nb_max );
28     return false;
29 }
30
31 int main() {
32     string mot = "chat";
33     if (partie(mot))
34         cout << "Félicitation, le mot est: " << mot << endl;
35     else
36         cout << "Désolé, le jeu est fini." << endl;
37     return 0;
38 }
```

ligne 21 : `nb_essais + 1` ⟶ `nb_essais++;`
ligne 24 : `not motTrouve(mot, stock)` ⟶ `motTrouve(mot, stock)`
ligne 27 : `nb_essai <= nb_max` ⟶ `nb_essai < nb_max`