Université Paris Diderot **Langages à objets avancés**
M1 **Année 2016-2017**

TP n° 4 - Références

Exercice 1 Essayez de deviner ce qu'est censé afficher le code suivant, puis vérifiez.

```
int a = 2;
int b = 4;
int & ref1 = a;
int & ref2 = b;

ref1 -= ref2;
ref2 += ref1;
cout << "ref1 = " << ref1 << ", ref2 = "<< ref2 << endl;

a    -= b;
b    += a;
cout << "a = " << a << ", b = "<< b << endl;
```

Exercice 2 Proposez une fonction `countDown` dans les cas suivants, de sorte que le programme termine et affiche `5, 4, 3, 2, 1, 0, done !`

— cas 1 :

```
int main(){
    int n=5;
    while (n>0) countDown1(n)--;
    cout << n << ", done !" << endl;
    return 0;
}
```

— cas 2 :

```
int main(){
    int n=5;
    while (n>0) countDown2(n);
    cout << n <<  ", done !" << endl;
    return 0;
}
```

— cas 3 :

```
int main(){
    int n=5;
    while (n>0) countDown3(& n);
    cout << n << ", done !" << endl;
    return 0;
}
```

— cas 4 :

```
int main(){
    int n=5;
    while (countDown4(n));
    cout << n << ",done !" << endl;
    return 0;
}
```

— cas 5 :

```
int main(){
    while (countDown5(5)) cout <<  ", "
    cout << " done !" << endl;
    return 0;
}
```

Exercice 3 Etudiez le comportement du code suivant :

```
int f(int &x){
    int rep=x;
    x=0;
    return rep;
}
int g(int &x){
    int rep=x;
    x=0;
    return 2*rep;
}

int main(){
    int n;
    n=1;
    cout << f(n) << " + " << g(n) << " = "<< endl;
    n=1;
    cout << f(n) + g(n) << endl;
    return 0;
}
```

Exercice 4 (Tableaux et références)

1. Construisez un tableau contenant les deux entiers 0 et 1, puis définissez deux variables références liées à ces deux cases. Incrémentez ces variables et vérifiez que le tableau est modifié.

   Par contre, symétriquement, on ne peut pas faire la chose suivante en c++ :

   ```
   int x=0, y=1, z=2;
   int & t[]={x,y};
   x++;
   cout << t[0];
   t[0]=z;
   cout << x;
   t[0]=5;
   cout << x;
   ```

   Car les tableaux de références ont été interdits dans la spécification du langage. Dans la suite de cet exercice on tachera de passer l'obstacle de cette interdiction.

2. Définissez une classe `IntRef` d'objets qui contiennent une référence vers un entier. Vous aurez alors la possibilité d'écrire :

   ```
   int x=0,y=1;
   IntRef rx(x), ry(y);
   IntRef t[]={rx,ry};
   ```

3. Remarquez que c++ vous autorise à écrire directement :

   ```
   int x=0,y=1;
   IntRef t[]={x,y};
   ```

   (Implicitement le constructeur de paramètre entier est appelé, voir cours).
   Faites à présent en sorte de pouvoir également écrire :

   ```
   x++;
   cout << t[0];
   ```

4. Passons maintenant au problème de l'écriture de `t[0]=z`, pour la rendre possible il faut réécrire l'opérateur d'affectation.
   Sa signature est `IntRef& IntRef::operator=(const IntRef& y)`. Vérifiez que tout se passe alors comme attendu pour `t[0]=z`

5. Il nous reste à franchir le dernier obstacle, l'écriture de `t[0]=5`. Surchargez l'opérateur d'affectation en écrivant une méthode `IntRef& IntRef::operator=(const int y)`

6. Finalement on peut donc faire tout de même des "tableaux de références" : à partir de deux entiers `int x=0, y=1;` construisez un tableau de taille 3x3 dont les cases i, j de somme paire seront des références vers x et celles de somme impaires seront des références vers y. Vérifiez.