TP et labo de la semaine 7

1. Codez les fonctions `somme(tableau)` et `moyenne(tableau)` qui retournent respectivement la somme des éléments du tableau, et leur moyenne. Codez aussi une fonction de tests unitaires.
2. Codez la fonction `tabEgal(tableau1, tableau2)` qui retourne `vrai` si et seulement si les tableaux ont la même longueur et contenu. Codez aussi une fonction de tests unitaires.
3. On cherche une fonction `histogramme(t)` qui prend en paramètre un tableau d'entiers entre 0 et 4 et qui retourne un tableau h de 5 nombres tel que `h[n]` est le nombre de fois que `n` apparait dans le tableau `t`. Codez cette fonction et une fonction de tests unitaires.

   Pour les tests unitaires, on pourrait faire les tests unitaires en se servant de `tabEgal` comme ceci :

   ```
   assert( tabEgal( histogramme([3,2,2,3,0,2]), [1,0,3,2,0] ) );
   ```

   On pourrait aussi utiliser le truc suivant :

   ```
   assert( histogramme([3,2,2,3,0,2]) == "1,0,3,2,0");
   ```

   Cela fonctionne car l'opérateur == fait la conversion automatique de tableaux en texte (si une des deux opérandes est un texte). Il est vrai que le test n'arrivera pas à détecter le cas erroné où histogramme retourne le texte `"1,0,3,2,0"` plutôt que le tableau [1, 0, 3, 2, 0] mais nous pouvons nous en contenter car c'est plus simple et direct.

4. On cherche une fonction `unique(tableau)` qui prend en paramètre un tableau de nombres et qui retourne un nouveau tableau de nombres contenant les éléments du tableau en paramètre, dans le même ordre, mais où les valeurs dupliquées ont été éliminées. Donc l'appel,

   ```
   unique([3, 11, 5, 22, 11, 11, 1, 22, 9])
   ```

   doit retourner le tableau `[3, 11, 5, 22, 1, 9]`. Codez cette fonction sans utiliser la méthode `push` (c'est-à-dire qu'il faut créer le tableau résultat de la bonne longueur). Faites une autre version en utilisant la méthode `push`. Codez aussi une fonction de tests unitaires.