TP 4 : STL, modèles de classes

Exercice 1 :
Programmer la méthode du crible d'Ératosthène pour calculer les nombres premiers inférieurs à N. Ont utilisera une structure de données adaptée à ce problème.

Pour rappel, il s'agit de lister tous les entiers de 2 à N, puis de supprimer de la liste tous les éléments multiples de 2. Outre 2, le plus petit élément restant est 3. On supprime alors les multiples de 3. Outre 2 et 3, le plus petit élément restant est 5. On supprime ainsi les multiples de 5. On continue de même jusqu'à ce que toute la liste ait été parcourue. Le nombres restants sont les nombres premiers.

Exercice 2 :
On définit, pour x_0 un entier donné, la suite

$$x_{n+1} = \begin{cases} x_n/2 & \text{si } n \text{ est pair,} \\ 3x_n + 1 & \text{sinon.} \end{cases}$$

Il est conjecturé que pour tout x_0, cette suite finit par boucler sur les valeurs 1, 2 et 4. On veut vérifier cette conjecture pour tous les $x_0 \leq N$, avec N un entier donné. Écrire un programme qui fait cela. On remarquera que dès que la suite atteint une valeur x déjà atteinte par un x_0 précédent, on pourra arrêter le calcul.

Exercice 3 :
Définir un modèle (`template`) de classe `polynome`, pour représenter les polynômes à coefficients d'un certain type (on pourra utiliser `int` et `double`, ou bien encore les types `rationnel` ou `entier` écrits dans le précédent TP).

Cette classe devra contenir :

- Comme membres privés, un entier correspondant au degré du polynôme et un pointeur correspondant à l'adresse d'un tableau contenant les coefficients du polynôme (coefficients nuls y compris).
- Un constructeur prenant comme argument un tableau de coefficients dont on fournit la taille.
- Un constructeur prenant comme argument un seul coefficient, et créant un polynôme constant.
- Faut-il redéfinir à la main le constructeur par copie ? le destructeur ? l'opérateur `=` ? Si oui, le faire.
- Une surcharge des opérateurs `+`, `-`, `*` (comme fonctions amies) et `+=`, `-=`, `*=` (comme fonctions membres).
- Une surcharge de l'opérateur `<<` pour afficher un polynôme.
- Deux surcharges de l'opérateur `()` : une première qui évalue le polynôme en un point, une deuxième qui compose deux polynômes.