TP : reconnaissance de classes Θ (Début)

Rappel (simplifié)

On dit qu'un programme est d'une **complexité** dans $\Theta(f(n))$
si le nombre d'opérations qu'effectue ce programme est proportionnel à $f(n)$

Les programmes demandés ci dessous devont être écrits "sans tricher" :

- les répétitives doivent être entre 1 et n
  Une répétitive `Pour i de 1 à` 2^n est le type même de ce dont je ne veux pas
- les points d'arrêts et les appels initiaux des récursives ne doivent comporter ni exponentiel, ni puissance.

Question 1

1. écrire des programmes
   - *void* $f_1(int\ n)$ dont la complexité soit dans $\Theta(n)$,
   - *void* $f_3(int\ n)$ dont la complexité soit dans $\Theta(n^3)$,
   - *void* $g_2(int\ n)$ dont la complexité soit dans $\Theta(2^n)$
   - et *void* $g_3(int\ n)$ dont la complexité soit dans $\Theta(3^n)$.

   A partir de quelle valeur de n sentez vous la différence entre l'exécution des programmes
   - $f_1(n)$ et $f_3(n)$?
   - $f_3(n)$ et $g_2(n)$?
   - $g_2(n)$ et $g_3(n)$?

   y a-t-il une valeur de n pour laquelle vous sentez une différence entre
   - l'exécution du programme $f_3(n)$
     et l'exécution successive des programmes $f_1(n)$ et $f_3(n)$?
   - l'exécution du programme $g_2(n)$
     et l'exécution successive des programmes $f_3(n)$ et $g_2(n)$?
   - l'exécution du programme $g_3(n)$
     et l'exécution successive des programmes $g_2(n)$ et $g_3(n)$?

   Indications :

   Pour les programmes de complexité polynomiale, imbriquez des répétitives.
   Pour les programmes de complexité exponentielle, utilisez des appels récursifs.

Travail à rendre pour la question 1

Le fichier `Q1.cpp` devra contenir
- le code de toutes vos fonctions
- Un main qui réponde à chacune des six questions et pour chacune des questions
  - affiche de quelle question il s'agit
  - la valeur trouvée pour n
  - et fasse les deux appels correspondants en affichant un message au début de chacun des deux appels.
- et la dernière ligne du fichier devra être `/ / g++ Q1.cpp -o R1`