Sorbonne Université
Master 2 Ingénierie mathématique
Programmation en C++

Examen du 23 octobre 2019
Durée : 1 heure 30

— **Tous** *les fichiers rendus devront contenir vos* **nom et prénoms**.
— *Il est très fortement suggéré d'envoyer des fichiers bien organisés, et qui ne créent pas d'erreurs à la compilation.*
— *Les deux parties sont indépendantes entre elles.*

1 Marche aléatoire

On considère la marche aléatoire symétrique sur l'ensemble $E_N = \{0, \dots, N\}$: pour un $X_0 \in E_N$ donné, on définit une suite $(X_n)_{n\in\mathbf{N}}$ par la relation :

$$X_{n+1} = \begin{cases} X_n + 1 & \text{avec probabilité } 1/2, \\ X_n - 1 & \text{avec probabilité } 1/2. \end{cases}$$

On s'intéresse à la position X_τ au temps $\tau = \inf\{n \in \mathbf{N}, X_n \in \{0, N\}\}$. Intuitivement, τ est le premier temps d'atteinte du "bord" de E_N, X_τ est la position à laquelle on est sorti. On a presque sûrement $X_\tau \in \{0, N\}$.

Dans un fichier `MA.cpp`, écrire un programme qui pour chaque valeur de $n \in E_N$ (on prendra $N = 10$), simule la marche $(X_n)_{n\in\mathbf{N}}$ jusqu'au temps τ, et calcule à partir de cela une approximation de $\mathbf{P}(X_\tau = 0) = 1 - \mathbf{P}(X_\tau = N)$. On utilisera 10^6 simulations à chaque fois.

2 Développement limités

On va programmer dans cette partie une méthode permettant de manipuler les développements limités d'une fonction en 0. Un développement limité à l'ordre n est une expression de la forme

$$f(x) = f_0 + f_1 x + \ldots + f_n x^n + o(x^n).$$

À partir de développements limités de f et de g, on peut obtenir un développement limité de $f + g$, fg, f/g (si $g(0) \neq 0$), et de $f \circ g$ (si $g(0) = 0$).

Le but de cette partie est de représenter informatiquement de tels objets.

2.1 Ordre 1

On ne s'intéresse dans un premier temps qu'au cas de développements à l'ordre 1, de la forme $f(x) = f_0 + f_1 x + o(x)$. Noter que l'on a $f_0 = f(0)$ et $f_1 = f'(0)$.

1. Dans des fichiers `DL.cpp` et `DL.hpp`, écrire une classe `DL2`, contenant deux membres privés `f0` et `f1`, de type `double`, qui correspondront à la valeur de la fonction et celle de sa dérivée en 0 .

2. Écrire un constructeur pour cette classe, prenant en argument les deux valeurs à assigner à `f0` et `f1`. On fournira comme valeur par défaut `0` à la variable `f1`.

   Définir une variable globale constante notée `x` et correspondant au développement $0+x+o(x)$.

3. Surcharger l'opérateur `<<` pour permettre d'afficher un développement limité, sur le modèle de `1.5 + 2.1x`. On ne se souciera pas des affichages inhabituels en fonction des valeurs de `f0` et `f1`. Par exemple, on ne cherchera pas à éviter les affichages tels que `0 + -3.1x` ou `2.5 + 0x`.

4. Écrire des surcharges des opérateurs `+`, `-`, `/` et `*`, correspondant aux opérations attendues sur les développements limités. Plus précisément, l'addition et la soustraction ont lieu terme à terme ; la multiplication et la division sont données par :

$$(a_0 + a_1x + o(x))(b_0 + b_1x + o(x)) = a_0b_0 + (a_0b_1 + a_1b_0)x + o(x)$$

   et

$$\frac{a_0 + a_1x + o(x)}{b_0 + b_1x + o(x)} = \frac{a_0}{b_0} + \frac{a_1b_0 - a_0b_1}{b_0^2}x + o(x)$$

5. Écrire un programme `fraction.cpp` qui donne la valeur numérique de la dérivée de $\varphi : t \mapsto \frac{t+t^3}{1+t+t^5}$ au point $t = 0.5$ (pour cela, faire un développement limité de $\varphi(0.5 + x)$).

2.2 Ordre n

Dans cette partie, on s'intéresse maintenant à des développements d'ordre quelconque.

1. Écrire une classe `DL`, contenant un membre privé `ordre` de type `int` et un autre nommé `coefs` de type `std::vector<double>` correspodant respectivement à l'ordre jusqu'auquel le développement limité est fait, et la liste des coefficients. Par exemple, le développement limité $1 + x^2/2 - 3x^3 + o(x^4)$ serait stocké dans une variable pour laquelle le membre `ordre` vaudrait `4`, et le membre `coefs` contiendrait les valeurs `1`, `0`, `0.5`, `-3` et `0`.

2. Écrire deux constructeurs pour cette classe : un premier prenant en argument un vecteur à recopier dans `coef`, un deuxième prenant en argument un `double` et un `int`, qui correspond au développement limité $\alpha + o(x^n)$ (on donnera par défaut la valeur 100 à n).

3. Surcharger l'opérateur `<<` pour permettre d'afficher un `DL`, sur le même modèle qu'en partie 2.1.

4. Écrire des surcharges des opérateurs `+`, `-` et `*`, faisant les opérations usuelles sur les développements limités. La multiplication peut se faire en sommant tous les $a_ib_jx^{i+j}$ jusqu'à l'ordre adéquat (ne pas oublier que $x^no(x^m) = o(x^{n+m})$ et que $o(x^n) + o(x^m) = o(x^n)$ si $n \leq m$).

5. Surcharger l'opérateur `()` pour calculer la composée de deux développements limités. Pour le calcul du développement de $f \circ g$, on ajoutera successivement des puissances du développement de g.

6. À l'aide de la multiplication et de la composition de développement limités, écrire le code d'une surcharge de `/` pour la division. On commencera par calculer l'inverse du dénominateur avec la formule $(1 - x)^{-1} = 1 + x + x^2 + \ldots + x^n + o(x^n)$, puis on fera un produit.

7. Écrire un programme `tan.cpp` qui calcul un développement limité à l'ordre 10 de tan = sin / cos en 0.