Université de Pau et des Pays de l'Adour **Année 2017-2018**
Département de Mathématiques

L2-Info–Calcul scientifique
TD n^o 6

Exercice 1

1. Ecrire une classe ‘‘`IntegrationFormula`’’ permettant de manipuler des définir des formules d'intégration.
   Cette classe contiendra:
   Les attributs privés:
   `double _a,_b; représentant les bornes de l'intervalle d'intégration`.
   int _n; le nombre de sous-découpages de l'intervalle [_a,_b].
   Les constructeurs (au moins ceux-là):
   `IntegrationFormula();` constructeur sans paramètre
   `IntegrationFormula(double a, double b, int n);` constructeur initialisant _a,_b et _n aux valeurs de a, b et n respectivement
   les fonctions membres publiques :
   `double getA() const;` renvoyant _a
   `double getB() const;` renvoyant _b
   `int getN() const;` renvoyant _n
   `void setABN(double a, double b, int n);` initialisant _a, _b, _n.
   `double computePointsMilieux(double (*f) (double)) const;` effectuant le calcul approché de $\int_a^b f(x)dx$ par la formule des Points Milieux
   `double computeTrapeze(double (*f) (double)) const;` effectuant le calcul approché de $\int_a^b f(x)dx$ par la formule des Trapèzes
   `double computeSimpson(double (*f) (double)) const;` effectuant le calcul approché de $\int_a^b f(x)dx$ par la formule de Simpson
   void computeErrors(double (*f)(double), double exact_value) `const` ; Calculant et affichant à l'écran, pour les trois méthodes ci-dessus l'erreur d'approximation.
   Les opérateurs d'entrée/sortie
   $<<$ écrivant _a, _b et _n.
   $>>$ lisant _a, _b et _n.

2. Ecrire un programme principal, ainsi qu'une fonction f permettant de tester la classe écrite ci-dessus et notamment les trois fonctions de calcul approché d'une intégrale.
   On prendra pour f la fonction $f(x) = sin(x)$ que l'on intègrera sur $[0, \pi]$.
   On testera également la fonction membre computeErrors()

Exercice 2:
On cherche à créer une version de la classe précédente beaucoup plus générale pour pouvoir l'utiliser pour une formule quelconque et pas seulement pour les trois formules de l'exercice 1.
Reprendre la classe IntegrationFormula ci-dessus pour en faire une nouvelle version en suivant les instructions suivantes:

1. Rajouter les attributs suivants:
   int _k; nombre de noeuds de la formules
   std::vector<double> _nodes; vecteur contenant les noeuds de la formule sur [0,1].
   std::vector<double> _weigths; vecteur de poids de la formule.
   string _name; // nom de la formule

2. Changer le constructeur `IntegrationFormula(double a, double b, int n)`, en : `IntegrationFormula(double a, double b, int n, const std::vector<double>& nodes, const std::vector<double>& w)` permettant d'initialiser également les vecteurs _nodes et _weigths.

3. Rajouter la fonction void setNodesAndWeigths(`const std::vector<double>& nodes`, `const std::vector<double>& w`) permettant d'initialiser les poids et les noeuds de la formule.

4. Éliminer les fonctions `computePointsMilieux(), computeTrapeze(), computeSimpson()` et écrire une seule fonction de calcul:
   `double computeIntegral(double (*f) (double)) const;`

5. Modifier la fonction computeErrors() pour qu'elle affiche à l'écran, le nom de la méthode ainsi que l'erreur commise pour cette méthode.

6. Refaire un programme principal pour tester cette version de la classe IntegrationFormula. On testera sur la formule de Simpson avec la même fonction f que dans l'exercice précédent.
   De plus, on testera également avec la formule de Boole dont les noeuds sont: 0, 1/4, 1/2, 3/4, 1 et les poids correspondants: 7/90, 32/90,12/90,32/90,7/90.