Ce TP a pour objectif de mettre en pratique l'interpolation d'un nuage de points par différences divisées. Pour ce faire, vous commencerez par récupérer sur Sakai le fichier `tpInterpolation.zip` que vous décompresserez dans votre répertoire personnel. Ce fichier contient une bibliothèque `nuageDePoints` pour la gestion (lecture/écriture, parcours, ajout) d'une liste chaînée de points ainsi qu'une bibliothèque `interp` qui définit une structure `Polynome` ainsi que les prototypes des fonctions nécessaires à l'interpolation.

EXERCICE 1: INTERPOLATION POLYNÔMIALE PAR DIFFÉRENCES DIVISÉES

Dans cet exercice, vous **complèterez** le fichier `interp.c` pour implémenter les différentes fonctions déclarées dans `interp.h`.

1. **Évaluation d'un polynôme par schéma de Horner :** Implémentez la fonction `double pEval( Polynome p, double x )` qui calcule la valeur d'un polynôme p en x :

$$p(x) = \sum_{i=0}^{N} a_i \prod_{j=1}^{i} (x - c_{j-1}) = a_0 + (x - c_0)\left\{a_1 + (x - c_1)\left[a_2 + (x - c_2)\big(\ldots(a_{N-1} + (x - c_{N-1})a_N)\big)\right]\right\}$$

   en implémentant l'algorithme de l'exercice 5.c du TD1 (rappelé ci-dessous).
   `Polynome` est une structure contenant un tableau `A` des $N+1$ coefficients du polynôme p et un tableau `C` des $N+1$ centres (racines) du polynôme p.

   fonction *pEval(p : Polynome, x : réel)*
   - **Variables** : pX : réel
     n : entier
   - pX ← p.A[p.N]
   - n ← p.N-1
   - **tant que** *n ≥ 0* **faire**
     - pX ← pX * (x - p.C[n]) + p.A[n]
     - n ← n - 1
   - **retourner** pX

   Soit $T = \{\mathbf{X}, \mathbf{Y}\}$ un nuage de $N+1$ points, avec $\mathbf{X} = [X_0, \ldots, X_N]$ un vecteur de points réels distincts deux à deux et $\mathbf{Y} = f(\mathbf{X})$ un vecteur d'échantillonnage de la fonction f aux points $\mathbf{X}$.

2. **Différences divisées :** Complétez la fonction `Polynome diffDiv( NuagePts* nPts )` qui calcule le polynôme d'interpolation par différences divisées à partir d'un nuage de points (algorithme vu en cours).

3. **Interpolation polynômiale :** Complétez la fonction `NuagePts interpol( NuagePts nPts, int M, double* xI )` qui détermine le polynôme d'interpolation par différences divisées de $f(x)$ puis calcule les valeurs $\mathbf{Y}^I$ de ce polynôme pour les M points du vecteur $\mathbf{X}^I = [X_0^I, \ldots, X_M^I]$. Ces valeurs seront conservées dans une variable de type `NuagePts` qui sera renvoyée comme résultat de la fonction.

EXERCICE 2: APPLICATION DE L'INTERPOLATION POLYNÔMIALE

1. Utiliser la fonction de l'exercice 1.3 pour interpoler les nuages de points donnés dans les exercices 1, 2 et 4 du TD2.
2. Dans le cas de l'exercice 4, comparer la courbe avec la fonction exacte $f(x) = \sin\left(\frac{x}{2}\right) e^{-0.1x}$ dans l'intervalle $[0, 60]$.

   Dans le programme principal, on sauvegardera les valeurs de $\mathbf{X}^I$ et $\mathbf{Y}^I = p_N(\mathbf{X}^I)$ dans un fichier texte que l'on nommera interpolation.txt. On pourra utiliser la fonction `void ecrireNuagePts( char* nomFichier, NuagePts* liste )` de la bibliothèque `nuageDePoints.h`. L'affichage devra être du type :

   0.1 2.212
   0.2 2.534
   0.3 3.123
   ...

   Pour permettre une vérification graphique de la fonction d'interpolation, sauvegardez dans un autre fichier texte (nommé echantillon.txt) les vecteurs d'entrée $\mathbf{X}$ et $\mathbf{Y}$.
   Vous pourrez utiliser le script Matlab `dispResC.m` pour cette visualisation.