Laboratoire no. 4

Objectif

Réaliser un laboratoire de synthèse, c'est-à-dire faisant appel aux concepts de classe, généricité et exception.

Donnée

Soit le code (partiel) suivant :
(code source disponible dans :
Cyberlean, section "Laboratoires", fichier main_labo_04.cpp)

```
#include <cstdlib>
#include <iostream>
#include <list>
#include <vector>
#include "collection_g.h"
#include "exceptions.h"
#include "produit.h"

using namespace std;

int main() {

   {
      cout << "------------------------------------------------------------" << endl;
      cout << "Test sur Collection<char, vector> :" << endl;
      try {
         Collection<char, vector> c;
         for (char ch = 'A'; ch < 'D'; ++ch)
            c.ajouter(ch);
         cout << c << " (taille = " << c.taille() << ")" << endl;
         c.get(0) = 'B';
         c.get(1) = c.get(2);
         c.get(2) = 'D';
         cout << c << " (taille = " << c.taille() << ")" << endl;
         cout << boolalpha
              << c.contient('A') << endl
              << c.contient('D') << endl
              << noboolalpha;
         c.vider();
         cout << c << " (taille = " << c.taille() << ")" << endl;
         cout << c.get(0) << endl;
      } catch (const IndiceNonValide& e) {
         cout << e.what() << endl;
      }
      cout << "------------------------------------------------------------" << endl;
      cout << endl;
   }
```

```
   {
      cout << "-----------------------------------------------------------" << endl;
      cout << "Test sur Produit :" << endl;
      try {
         // un produit se caractérise par un no, un libellé, un prix
         Produit p1(1, "Produit 1", 0.05);
         cout << p1 << endl;
         {
            try {
               Produit p2(2, "Produit 2", 0);
            } catch (const PrixNonValide& e) {
               cout << e.what() << endl;
            }
         }
         p1.setPrix(0.0);
      } catch (const PrixNonValide& e) {
         cout << e.what() << endl;
      }
      cout << "-----------------------------------------------------------" << endl;
      cout << endl;
   }

   {
      cout << "-----------------------------------------------------------" << endl;
      cout << "Test sur Collection<Produit, list> :" << endl;
      try {
         Collection<Produit, list> c;
         Produit p1(1, "Produit 1", 1.55);
         Produit p2(2, "Produit 2", 5);
         c.ajouter(p1);
         c.ajouter(p2);
         cout << c << " (taille = " << c.taille() << ")" << endl;
         Produit tmp = c.get(0);
         c.get(0) = c.get(1);
         c.get(1) = tmp;
         cout << c << " (taille = " << c.taille() << ")" << endl;
         cout << boolalpha
              << c.contient(p1) << endl
              << c.contient(p2) << endl
              << noboolalpha;
         {
            < à compléter 1 >
            // On parcourt la collection en majorant le prix de chacun
            // des produits de 10%
            c.parcourir(< à compléter 2 >);
            cout << c << " (taille = " << c.taille() << ")" << endl;
         }
         c.vider();
         cout << c << " (taille = " << c.taille() << ")" << endl;
      } catch (const IndiceNonValide& e) {
         cout << e.what() << endl;
      }
      cout << "-----------------------------------------------------------" << endl;
      cout << endl;
   }

   return EXIT_SUCCESS;
}
```

On demande ici :

- d'implémenter les modules *collection_g*, *exceptions* et *produit*
- de compléter les parties notées *<à compléter 1>* et *<à compléter 2>* du *main*

de telle sorte que le code produise à l'exécution les résultats suivants :

```
------------------------------------------------------------
Test sur Collection<char, vector> :
[A, B, C] (taille = 3)
[B, C, D] (taille = 3)
false
true
[] (taille = 0)
Erreur dans Collection::get :
n doit etre strictement plus petit que collection.size()
------------------------------------------------------------

------------------------------------------------------------
Test sur Produit :
(1, "p", 0.05)
Erreur dans Produit::Produit :
le prix doit etre >= 5 cts !
Erreur dans Produit::setPrix :
le prix doit etre >= 5 cts !
------------------------------------------------------------

------------------------------------------------------------
Test sur Collection<Produit, list> :
[(1, "Produit 1", 1.55), (2, "Produit 2", 5.00)] (taille = 2)
[(2, "Produit 2", 5.00), (1, "Produit 1", 1.55)] (taille = 2)
true
true
[(2, "Produit 2", 5.50), (1, "Produit 1", 1.71)] (taille = 2)
[] (taille = 0)
------------------------------------------------------------
```

IMPORTANT

- Le code du programme fourni plus haut ne doit PAS être modifié (à l'exception bien sûr des deux parties < à compléter >).
- N'implémenter que le code strictement nécessaire à la résolution du problème.
- Le cas d'un défaut de mémoire lors de l'ajout d'un élément dans la collection n'est pas à prendre en compte ici.

A réaliser

☐ Seul
☒ Par groupe de 3
(Inscription des groupes dans : Teams / INF2-A-RRH / General / Fichiers / Groupes_labo_4.xls)

Travail à rendre **vendredi soir 24 avril (minuit)**

☒ Listing[1] du code (.pdf) + fichiers sources (.h et .cpp) dans :
\\eistore1\cours\tic\RRH\INF2\Rendus\<votre répertoire>\Labo_4
où <votre répertoire> = répertoire du membre du groupe venant en premier dans l'ordre alphab.

[1] L'impression papier de listings de vos fichiers souces (.h et .cpp) n'étant plus d'actualité, il vous est demandé de nous fournir, en lieu et place, un unique fichier pdf contenant une copie du code de l'ensemble des fichiers sources (.h et .cpp) de votre projet.
La procédure à suivre pour produire ledit fichier pdf est décrite dans le document "Comment créer un listing d'un projet", disponible sur Cyberlearn, section "Laboratoires".