TP POO/C++ : Gestion des exceptions

1a . exo_1a.cpp : Dans la fonction Exo_1(), initialiser une variable entière non nulle numerateur, et lire au clavier le dénominateur denominateur, lui aussi entier. Tester le dénominateur :

- s'il est non nul, afficher le résultat de la division entière de numerateur par denominateur (opérateur /).
- s'il est nul, lever (throw) une exception entière (la valeur du denominateur par exemple).

Sécuriser l'ensemble du corps de la fonction Exo_1() en l'incluant dans une instruction try-catch. Dans le traitement de l'exception capturée (catch), afficher la valeur de l'exception dans le flux d'erreur (cerr).

Remarque : la valeur entière de l'exception est ici « vue » comme un **code d'erreur**.

1aa . exo_1aa.cpp : Modifier le programme précédent pour lever une exception "Division par zero" de type string. Dans le traitement correspondant, afficher le message associée à l'exception levée.

Remarque : la chaîne de caractère de l'exception sera donc « vue » comme un **message d'erreur**.

1b . exo_1b.cpp : Modifier le programme précédent exo_1a.cpp pour que dans le traitement d'exception (catch), on relance l'exception (throw) après l'affichage. Modifier le corps de la fonction main() pour permettre la capture de toute exception quelle qu'elle soit. Dans le traitement correspondant, l'exception ne pouvant être identifiée, afficher simplement le message : Exception interceptée !

1c . exo_1c.cpp : Modifier le programme précédent pour lever et traiter une exception range_error. Modifier le traitement d'exception pour qu'il relance cette exception après l'affichage. Modifier le corps de la fonction main() pour permettre la capture d'une exception range_error. Dans le traitement correspondant, afficher le message associée à l'exception range_error.

2 . exo_2.cpp : Dans la fonction Exo_2(), définir une variable caractère c, et, dans une boucle infinie, lire au clavier un caractère. Ne sortir de la boucle, en levant une exception CEMinus, que si c n'est pas une minuscule (cf. islower). Il faudra bien entendu utiliser un bloc try-catch, même si le traitement d'exception est vide. Vous devez ajouter une classe CEMinus par dérivation publique de exception. Capturer cette exception et afficher son contenu (toujours par la fonction héritée what()).

