Exponential

Implementieren Sie die Klassen **`Exponential`** zur Durchführung von mathematischen Operationen mit Zahlen in Exponentialdarstellung (also in der Form m^n, wobei m und n ganze Zahlen sind). Die Multiplikation zweier Objekte der Klasse **`Exponential`** ist erlaubt, falls die beiden Basen oder die beiden Exponenten gleich sind. Das Ergebnis ist wieder ein Objekt der Klasse **`Exponential.`**

Berücksichtigen Sie folgende Sonderfälle:

• 0^0 ist nicht erlaubt. Sichern Sie Ihre Methoden entsprechend ab, so dass dieser Fall nicht eintreten kann (Werfen Sie gegebenenfalls eine Exception vom Typ runtime_error).

• 0^n ist für n > 0 gleich 0 und sollte intern immer als 0^1 dargestellt werden.

• 0^n ist für n < 0 unendlich und sollte intern immer als 0^{-1} dargestellt werden.

• 1^n ist (für alle n) gleich 1 und sollte intern immer als 1^0 dargestellt werden.

• n^0 ist (für n!=0) gleich 1 und sollte intern immer als 1^0 dargestellt werden.

Defaultwert für Basis und Exponent soll 1 sein.

Folgende Operatoren und Methoden sind zu implementieren:

- Konstruktor(en): Ein Konstruktoraufruf soll mit zwei, einem oder keinem int-Parameter möglich sein. Defaultwert für nicht übergebene Parameterwerte soll 1 sein. Der erste Parameter gibt jeweils die Basis an, der zweite den Exponenten. Somit ergibt Exponential{-2,3} -2^3, Exponential{5} 5^1 und Exponential{} 1^1 (dieses ist allerdings intern als 1^0 darzustellen).
- Operator *: Die Multiplikation zweier Exponentiale ist möglich, falls die beiden Werte in der Basis oder im Exponenten übereinstimmen. Es gelten die Rechenregeln:
  $a^x * a^y = a^{x+y}$
  $a^x * b^x = (ab)^x$
  Sind sowohl Basis als auch Exponent gleich, so ist die erste Regel anzuwenden. Kann die Multiplikation nicht durchgeführt werden, so ist eine Exception vom Typ runtime_error zu werfen.
- Operator <<: Ein Exponential wird ausgegeben, in der Form Basis^Exponent z.B.: -1^7
- Zusatz für 15 Punkte: Die Methode **`bool basiswechsel(int b)`** soll die Basis des Exponentials, falls möglich in den Wert b ändern (damit der Wert gleich bleibt, ist natürlich auch der Exponent entsprechend anzupassen). Die Methode liefert **`true`**, falls die Änderung durchgeführt wurde und **`false`** sonst. Die Basis kann auf b gesetzt werden, wenn entweder die Basis des Exponentials eine Potenz von b ist oder b eine Potenz der Basis des Exponentials und die Division des Exponenten ganzzahlig durchführbar ist, z.B. x^y in Basis b:
  - falls $x=b^a$: $x^y = (b^a)^y = b^{a*y}$
  - falls $b=x^a$ und y/a ist ganz: $x^y = (x^a)^{y/a} = b^{y/a}$
    (Anmerkung: Ob eine Zahl eine Potenz einer anderen Zahl ist, lässt sich durch fortgesetztes Dividieren prüfen.)
- Zusatz für 10 Punkte : Die Methode **`vector<Exponential> faktoren() const`** soll Faktoren, die Potenzen von 2, 3, 5 oder 7 sind, von einem Exponential abspalten und die so ermittelten Faktoren in der Reihenfolge aufsteigender Basen in einem Array retournieren, z.B.: $1430^2 = 2^2 * 5^2 * 143^2$, d.h. für das Exponential 1430^2 wird ein Vektor mit den Exponentialen 2^2, 5^2 und 143^2 retourniert. Beachten Sie, dass ein Faktor auch mehrmals auftreten kann, dann ist geeignet zusammenzufassen, z.B.: $12^2 = 2^2 * 2^2 * 3^2 = 2^4 * 3^2$.

Implementieren Sie die Klasse **`Exponential`** mit den notwendigen Konstruktoren, Methoden und Operatoren, sodass jedenfalls das Rahmenprogramm (exponential.h, exponential.cpp und testexponential.cpp verfügbar unter /home/Xchange/ue12) kompiliert und ausgeführt werden kann und die gewünschten Ergebnisse liefert. Achten Sie in Ihren Konstruktoren darauf, dass nur gültige Objekte erstellt werden können. Werfen Sie gegebenenfalls eine Exception vom Typ **`runtime_error`**.

Für Ihr Programm dürfen Sie **nur** die im vorgegebenen Rahmenprogramm angeführten include-Dateien verwenden!

Instanzvariablen sind **`private`** zu definieren und die Verwendung globaler Variablen ist (abgesehen von im Rahmenprogramm eventuell bereits definierten) nicht erlaubt! Interpretationsspielraum in der Angabe können Sie zu Ihren Gunsten nutzen.

Die Teilaufgaben, bei denen keine Punkteanzahl angegeben ist, gelten als Basisfunktionalität. Für eine positive Beurteilung ist zumindest die Basisfunktionalität zu implementieren. Diese wird mit 30 Punkten bewertet.

Die übrigen Teilaufgaben müssen nicht unbedingt implementiert werden, führen aber im Falle einer korrekten Implementierung zu einer entsprechenden Erhöhung der Punkteanzahl.