PR1 VU Programmierung 1	Klausurvorbereitung	

Pizzabude

Implementieren Sie die Klassen `Topping` und `Pizza`:

Ein `Topping`-Objekt hat die Instanzvariablen `name` (`string`, nicht leer), `allergenes` (`string`, eventuell leer) und `specialty` (`bool`). Die Variable `allergenes` (Allergenkennzeichnung) enthält die zutreffenden Codes für die im Topping (= Belag) enthaltenen Allergene in Form eines Strings. Mögliche Codes sind Buchstaben zwischen 'A' und 'R' (jeweils inklusive), nicht aber 'I, 'J', 'K' oder 'Q'. Die einzelnen Codes sind innerhalb des Strings alphabetisch aufsteigend sortiert. Sie können der Einfachheit halber davon ausgehen, dass alle Allergenkennzeichnungen, die an Ihre Funktionen als Parameter übergeben werden, korrekt sind (also nur gültige Codes in richtiger Reihenfolge enthalten). Wenn eine Ihrer Funktionen eine Allergenkennzeichnung retourniert, ist aber sicherzustellen, dass der retournierte Wert nach den obigen Regeln gültig ist. Für die Klasse `Topping` sind folgende Methoden und Funktionen zu implementieren:

- Konstruktor(en) mit 1, 2 oder 3 Parametern: Name, Allergenkennzeichnung und Spezialität in dieser Reihenfolge. Allergenkennzeichnung und Spezialität sind optional und per Default gleich dem Leerstring bzw. `false`. Sollte einer der Parameter nicht die Voraussetzungen erfüllen (z. B. Name ist leer), ist eine Exception vom Typ `runtime_error` zu werfen.
- `bool is_specialty() const`: Retourniert den Wert der Instanzvariable `specialty`.
- `operator==`: Vergleicht zwei Beläge. Zwei Beläge sind gleich, wenn ihre Namen gleich sind.
- `operator<<`: Ein Belag muss in der Form *name [(Allergencodes)]* ausgegeben werden. Dabei bedeuten die eckigen Klammern, dass dieser Teil der Ausgabe optional ist. Das heißt: Die runden Klammern und die Liste der Allergencodes sind nur auszugeben, wenn die Allergenkennzeichnung des Belags (`this`-Objekts) nicht leer ist, z. B. **Frutti di mare (A, B, D, R)** oder, bei leerer Allergenkennzeichnung z. B. **Funghi**.

Ein `Pizza`-Objekt hat eine Instanzvariable `toppings`, eine Liste der Beläge, die auf der Pizza sind. Für die Klasse `Pizza` sind folgende Methoden und Funktionen zu implementieren:

- Konstruktor mit 0, 1 oder 2 Parametern: Liste der gewünschten Beläge und Liste der unerwünschten Beläge in dieser Reihenfolge. Defaultwert ist für beide Listen die leere Liste. Beim Erstellen eines neuen `Pizza`-Objekts werden zunächst die beiden Standardbeläge `Tomato sauce` und `Cheese` (beide Beläge sind keine Spezialitäten, die Allergenkennzeichnung für Tomatensauce ist leer, für Käse ist sie "G") in dieser Reihenfolge in die Liste der Beläge des neuen Objekts übernommen. Ein Standardbelag, der sich in der zweiten Parameterliste (Liste der unerwünschten Beläge) befindet, wird dabei nicht übernommen. Dann werden die Einträge aus dem ersten Parameter (Liste der gewünschten Beläge) unter Beibehaltung der Reihenfolge in die Liste der Beläge des neuen Objekts übernommen, wobei Mehrfacheinträge ignoriert werden. Jeder Belag darf im Endresultat also höchstens einmal auftreten. Der Konstruktor wirft nur dann eine Exception vom Typ `runtime_error`, wenn das zu erzeugende `Pizza`-Objekt gar keinen Belag mehr hätte.
- `double price() const`: Retourniert den Preis der Pizza (`this`-Objekt), der folgendermaßen berechnet wird: Eine Pizza mit den beiden Standardbelägen und zwei weiteren frei gewählten Belägen kostet 7 €. Das ist auch der Mindestpreis einer Pizza. Für jeden abgewählten Standardbelag kann ein Belag frei gewählt werden, ohne die Kosten zu erhöhen. Für jeden zusätzlichen Belag wird 1 € verrechnet und unabhängig davon werden für jeden Belag, der als Spezialtiät gekennzeichnet ist, weitere 0.5 € zum Preis dazugeschlagen.
- `operator<<`: Die Ausgabe eines Objekts vom Typ `Pizza` muss in der Form [{*Liste der Zutaten*}, *price* Euro] erfolgen, wobei *price* der mittels der Methode `price` ermittelte Preis der Pizza ist, z. B.: [{Tomato sauce, Cheese (G)}, 7 Euro]
- Zusatz für 10 Punkte: Erweitern Sie die Klasse `Pizza` um die Methode `string allergenes() const`:
  Diese Methode liefert eine Allergenkennzeichnung für die gesamte Pizza, indem die Codes für alle Beläge gesammelt werden und der Code "A" (für den Pizzateig) hinzugefügt wird. Der resultierende String muss den oben definierten Regeln für eine Allergenkennzeichnung genügen. (Tipp: in der Schleife `for (char c{'A'}; c<'S'; ++c)` durchläuft c alle Buchstaben zwischen 'A' und 'R' inklusive der Grenzen. Eine Methode in `Topping` mit einem `char` Parameter, die retourniert, ob der Belag das entsprechende Allergen enthält, wäre hilfreich.)
- Zusatz für 15 Punkte: Erweitern Sie die Klasse `Pizza` um die Methode
  `void accommodate(const string& forbidden)`: Es sind alle Beläge von der Pizza zu entfernen, die ein Allergen enthalten, das in `forbidden` vorkommt. Sie können davon ausgehen, dass `forbidden` den obigen Regeln für eine Allergenkennzeichnung genügt. Die relative Reihenfolge der übrig bleibenden Beläge ist beizubehalten. Sollte kein Belag übrig bleiben, die Pizza nach Entfernen der Beläge also "leer" sein, so ist der ursprüngliche Zustand (vor Aufruf der Methode) wieder herzustellen und eine Exception vom Typ `runtime_error` zu werfen.

Implementieren Sie die Klassen `Topping` und `Pizza` mit den notwendigen Konstruktoren, Methoden und Operatoren, sodass jedenfalls das Rahmenprogramm 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! Die Datenkapselung darf nicht durchbrochen werden. Es ist daher unter anderem nicht erlaubt, Referenzen oder Pointer auf private Instanzvariablen einer Klasse nach außen zu vermitteln, `friend`-Deklarationen (mit Ausnahme bei Operatorfunktionen) zu verwenden, oder setter-Methoden zu implementieren, die die Integrität der Daten nicht gewährleisten. 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.