PR1 VU Programmierung 1	Klausurvorbereitung	

Implementieren Sie die Klassen `Spieler` und `Mannschaft` zur Verwaltung eines Fußballturniers.

Ein `Spieler` hat einen Namen (`string` mit Länge > 0), eine Transfersumme (`int`-Wert in Mio €, ≥ 10 und ≤ 500) und eine Position, welche ein Wert aus der vordefinierten Enumeration `Position` (`Position::Tor`, `Position::Abwehr`, `Position::Mitte` und `Position::Sturm`) ist.

Für die Klasse `Spieler` sind folgende Methoden zu implementieren:

- Konstruktor(en) mit zwei bzw. drei Parametern. Name (`string`), Transfersumme (`int`) und Position in dieser Reihenfolge. Die Position ist optional und per Default `Position::Tor`. Sollte einer der übergebenen Werte nicht den Anforderungen entsprechen (z. B. Name mit Länge 0 oder Transfersumme nicht im erlaubten Bereich), so ist eine Exception vom Typ `runtime_error` zu werfen.
- `bool operator==(const Spieler&) const`: Der Einfachheit halber wird angenommen, dass die Spielernamen eindeutig sind. Diese Methode liefert daher `true`, wenn die Namen der beiden Spieler gleich sind, `false` sonst.
- `operator<<`: Ein Objekt des Typs `Spieler` muss in der Form [*Name*, *Transfersumme* Mio Euro, *Position*] ausgegeben werden. Der vordefinierte Vektor `positionsnnamen` kann für die Ausgabe der Enumerationswerte verwendet werden, z. B. `[Ronaldo, 120 Mio Euro, Sturm]`.

Eine `Mannschaft` hat einen Namen (`string` mit Länge > 0) und eine Spielerliste (`vector<Spieler>`). Für die Klasse `Mannschaft` sind folgende Methoden zu implementieren:

- Konstruktor(en) mit einem oder zwei Parametern, einem Namen (`string`) und einer Spielerliste (`vector<Spieler>`) in dieser Reihenfolge. Die Spielerliste ist optional. Wird sie nicht angegeben, so ist eine Mannschaft mit leerer Spielerliste zu erstellen. Sollte einer der übergebenen Werte nicht den Anforderungen entsprechen (z. B. Name mit Länge 0) oder in der Spielerliste ein Spieler mehrfach auftreten (gleicher Name), so ist eine Exception vom Typ `runtime_error` zu werfen.
- `vector<Spieler> engagieren(const vector<Spieler>&)`: Nimmt alle Spieler aus der als Parameter empfangenen Liste in die Mannschaft auf. Die neuen Spieler werden in der Reihenfolge, in der sie im Parameter auftreten, an das Ende der Spielerliste der Mannschaft angehängt. Sollte ein Spieler gleichen Namens bereits in der Mannschaft sein, so ist dieser nicht hinzuzufügen, sondern einfach zu überspringen. Zu retournieren ist die Liste der übersprungenen Spieler in der ursrprünglichen relativen Reihenfolge.
- `operator<<`: Die Ausgabe eines Objekts des Typs `Mannschaft` muss in der Form [*Name*, {*Liste der Spieler*}] erfolgen, zum Beispiel `[Barcelona, {[Messi, 180 Mio Euro, Sturm], [Umtiti, 60 Mio Euro, Tor]}]`.
- Zusatz für 10 Punkte:
  Erweitern sie die Klasse `Mannschaft` um eine Methode `string aufstellung() const`. Diese liefert die Mannschaftsaufstellung in Form eines Strings in der Codierung x-y-z (z. B. 4-4-2) zurück. Dabei ist x die Anzahl der Abwehrspieler, y die Anzahl der Mittelfeldspieler und z die Anzahl der Stürmer. Es ist darauf zu achten, dass es sich um eine gültige Aufstellung handelt, das heißt, es müssen 11 Spieler mit genau einem Tormann sein. Ist dies nicht der Fall, so ist eine Exception vom Typ `runtime_error` zu werfen. (Hinweis: Die Funktion `to_string()` kann zur Umwandlung von `int`-Werten in Strings verwendet werden.)
- Zusatz für 15 Punkte:
  Erweitern Sie die Klasse `Mannschaft` um eine Methode `vector<Spieler> entlassen(vector<Position>)`. Diese entfernt alle Spieler aus der Mannschaft, die eine der Positionen belegen, die im Parametervektor als Eintrag vorkommen. Die relative Reihenfolge der Spieler in der Mannschaft ist beizubehalten. Retourniert werden muss die Liste der entlassenen Spieler in der ursprünglichen relativen Reihenfolge.

Implementieren Sie die Klassen `Spieler` und `Manschaft` 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.