Упражнения: Ограничители за шаблонни класове

1. Универсална везна

Създайте клас Scale<T>, Съдържа два елемента: left и right. Получава ги чрез своя единствен конструктор:

- Scale(T left, T right)

Везната трябва да има един-единствен метод:

- T getHeavier()

По-големият от двата елемента е по-тежък. Методът трябва да връща **default(T)**, ако елементите са еднакви.

2. Подобрен списък

Създайте универсална структура данни, която може да съхранява произволен тип данни, който може да бъде сравняван. Реализирайте функциите:

- void Add(T element)
- T Remove(int index)
- bool Contains(T element)
- void Swap(int index1, int index2)
- int CountGreaterThan(T element)
- T Max()
- T Min()

Създайте команден интерпретатор, който чете команди и променя подобрения списък, който сте създали. Инициализирайте списъка да съхранява низове. Реализирайте командите:

- Add <element> - добавя даден елемент в края на списъка
- Remove <index> - премахва елемента, намиращ се на указаната позиция
- Contains <element> - отпечатва дали списъкът съдържа даден елемент (True или False)
- Swap <index> <index> - разменя местата на елементите с указаните индекси
- Greater <element> - преброява елементите, които са по-големи от подадения елемент и отпечатва техния брой
- Max - отпечатва максималния елемент от списъка
- Min - отпечатва минималния елемент от списъка
- Print - отпечатва всички елементи в списъка, всеки на отделен ред
- END - приключва с четенето на командите

Няма да има никакви невалидни команди във входните данни.

Примери

Вход	Изход
Add aa Add bb Add cc Max Min Greater aa Swap 0 2 Contains aa	cc aa 2 True cc bb aa

Print END	

3. Сортиране на подобрения списък

Разширете решението на предната задача чрез създаване на допълнителен клас Sorter. Той трябва да има един-единствен статичен метод Sort(), който може да сортира обекти от тип CustomList, съдържащи данни от произволен тип, който подлежи на сравняване. Разширете списъка с команди, така че да поддържа една допълнителна команда Sort:

- **Sort** - сортира елементите в списъка в нарастващ ред.

Примери

Вход	Изход
Add cc Add bb Add aa Sort Print END	aa bb cc

4. *Обхождане на подобрения списък

За всяка от командите за отпечатване вероятно сте използвали цикъл for. Разширете вашия подобрен списък, така че да реализира интерфейса IEnumerable<T>. Това би позволило да обхождата вашия списък с командата foreach.

Примери

Вход	Изход
Add aa Add bb Add cc Max Min Greater aa Swap 0 2 Print END	cc aa 2 cc bb aa

5. Tuple

Има нещо много странно в C#. Нарича се Tuple. Това е клас, който може да съхранява няколко обекта, но нека се фокусираме върху тип Tuple, който съхранява два обекта. Първият е "item1", а вторият - "item2". Това е нещо подобно на KeyValuePair с изключение на това, че просто съхранява елементи, които не са нито ключове, нито стойности. Странността идва от факта, че нямате никаква идея какво съдържат тези елементи. Името на класа нищо не ви подсказва, методите които има - също. И така, нека си представим, че по някаква причина бихме искали да се опитаме сами да направим такъв клас, ей така - просто за да упражним шаблоните.

Задачата: създайте клас "Tuple", съдържащ два обекта. Както споменахме, първият ще е "item1", а вторият - "item2". Тънкостта тук идва от това, да накараме класа да поддържа шаблони. Това ще рече когато създаваме нов обект

от клас "Tuple", трябва да начин изрично да укажем типа и на двата елемента поотделно.

Вход

Входните данни включват три реда:

- Първият съдържа името на човек и адресът му. Те са отделени с интервал(и). Вашата задача е да ги прочетете в tuple-а и да ги отпечатате на конзолата. Форматът на входните данни е:
<<first name> <last name>> <address>
- Вторият ред съдържа име на човек и количеството бира (int), което той може да изпие. Формат:
<name> <liters of beer>
- Последният ред съдържа Integer и Double. Формат:
<Integer> <Double>

Изход

- Отпечатайте елементите на tuple-а във формат: {item1} -> {item2}

Ограничения

Използвайте добрите практики, които сме учили. Създайте клас и му добавете getters и setters за клас-променливите му. Входните данни ще са валидни, няма нужда изрично да ги проверявате!

Пример

Вход	Изход
Sofka Tripova Stolipinovo Az 2 23 21.23212321	Sofka Tripova -> Stolipinovo Az -> 2 23 -> 21.23212321

6. Threeuple

Създайте клас **Threeuple**. Its name is telling us, that it will hold no longer, just a pair of objects. The task is simple, our **Threeuple** should **hold three objects**. Make it have getters and setters. You can even extend the previous class

Вход

Входните данни се състоят от три реда:

- Първият ред съдържа име, адрес и град, във формат:
<<first name> <last name>> <address> <town>
- Вторият ред съдържа име, литри бира и булева променлива със стойност **drunk** или **not**. Форматът е:
<name> <liters of beer> <drunk or not>
- Третият ред съдържа име, наличност по банковата сметка (double) и име на банката. Форматът е:
<name> <account balance> <bank name>

Изход

- Отпечатайте Threeuple обектите във формат: **{firstElement}** -> **{secondElement}** -> **{thirdElement}**

Примери

Вход	Изход
Sofka Tripova Stolipinovo Plovdiv MitkoShtaigata 18 drunk SashoKompota 0.10 NkqfaBanka	Sofka Tripova -> Stolipinovo -> Plovdiv MitkoShtaigata -> 18 -> True SashoKompota -> 0.1 -> NkqfaBanka
Ivan Ivanov Tepeto Plovdiv Mitko 18 not Sasho 0.10 NGB	Ivan Ivanov -> Tepeto -> Plovdiv Mitko -> 18 -> False Sasho -> 0.1 -> NGB

Бележки

Може да използвате и надградите решението на предната задача.