# Упражнения: Множества и речници

Тествайте решението в Judge: <https://judge.softuni.org/Contests/4160/05-Sets-and-Dictionaries-Basics>.

## **Брой на еднакви стойности в масив**

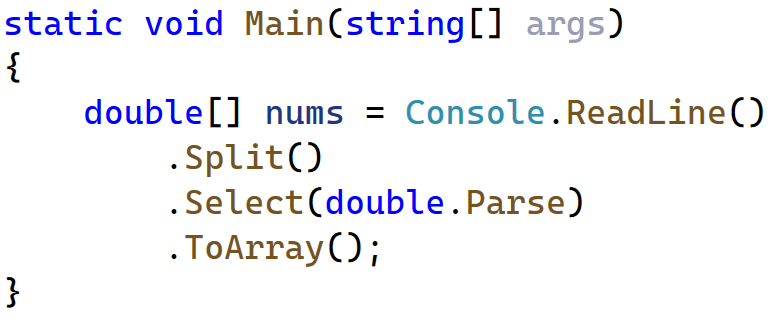
Напишете програма, която намира колко пъти едно число се среща в масив от реални числа.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| -2.5 4 3 -2.5 -5.5 4 3 3 -2.5 3 | -2.5 - 3 times  4 - 2 times  3 - 4 times  -5.5 - 1 times |
| 2 4 4 5 5 2 3 3 4 4 3 3 4 3 5 3 2 5 4 3 | 2 - 3 times  4 - 6 times  5 - 4 times  3 - 7 times |

#### **Насоки**

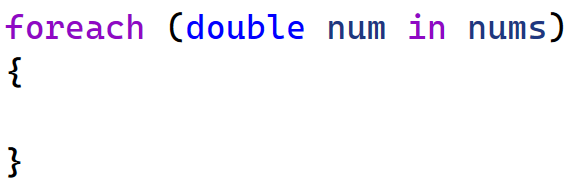
1. Прочетете **входа**:



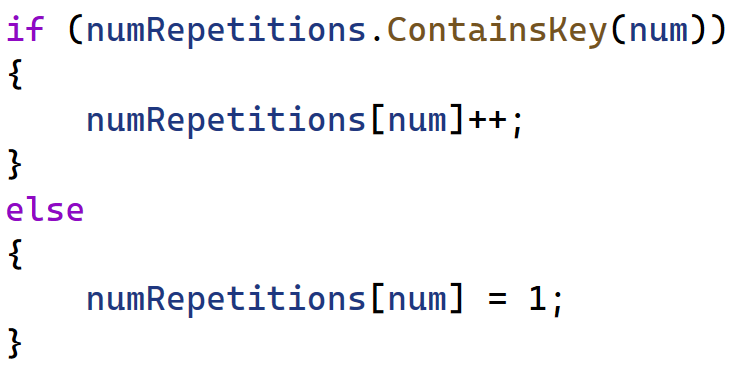
1. Създайте **речник**, който ще брои **повторенията** на всяко число:



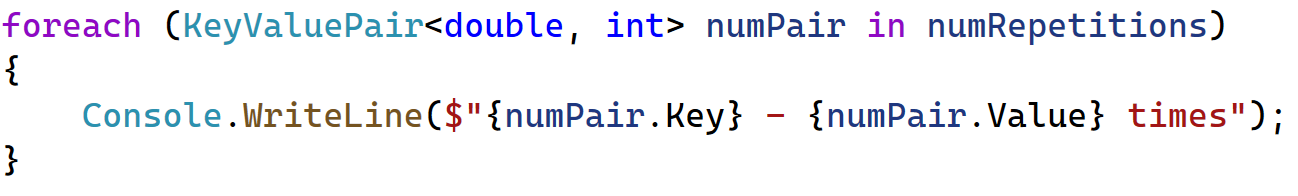
1. Създайте **foreach-цикъл**, който обхожда входа:



1. В него проверете дали числото **съществува** в **речника**. Ако е така, стойността му се **увеличава** с **едно**. В противен случай, **числото** се **добавя** със **стойност 1**:



1. Накрая отпечатайте числата чрез **foreach-цикъл**:



## Броя на еднакви символи в един низ

**Напишете програма, която намира броя на буквите в низ с изключение на (' ')**. Отпечатайте резултата в следния формат: **{буква} -> {брой пъти срещания}**

### Примери

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Вход** | **Изход** |
| text | t -> 2  e -> 1  x -> 1 | text text text | t -> 6  e -> 3  x -> 3 |

## **Средноаритметичен успех на ученици**

Напишете програма, която чете число **n**. След това получава **имената на n** на брой **ученици** с техните **оценки** и ги добавя към **бележниците им.**

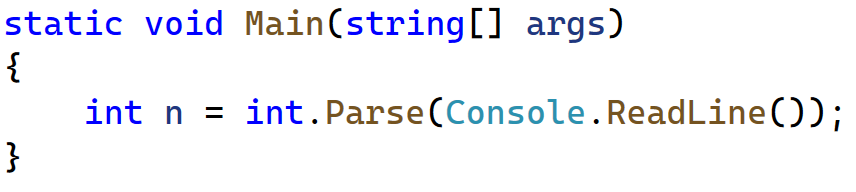
Накрая отпечатайте **името на всеки** **ученик** с неговите **оценки** и **средноаритметичен успех**. Всички оценки трябва да бъдат форматирани до **2 знака след десетичната точка**.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 7  Ivancho 5.20  Mariika 5.50  Ivancho 3.20  Mariika 2.50  Stamat 2.00  Mariika 3.46  Stamat 3.00 | Ivancho -> 5.20 3.20 (avg: 4.20)  Mariika -> 5.50 2.50 3.46 (avg: 3.82)  Stamat -> 2.00 3.00 (avg: 2.50) |
| 4  Vladimir 4.50  Petko 3.00  Vladimir 5.00  Petko 3.66 | Vladimir -> 4.50 5.00 (avg: 4.75)  Petko -> 3.00 3.66 (avg: 3.33) |
| 5  Gosho 6.00  Gosho 5.50  Gosho 6.00  Ivan 4.40  Petko 3.30 | Gosho -> 6.00 5.50 6.00 (avg: 5.83)  Ivan -> 4.40 (avg: 4.40)  Petko -> 3.30 (avg: 3.30) |

### Насоки

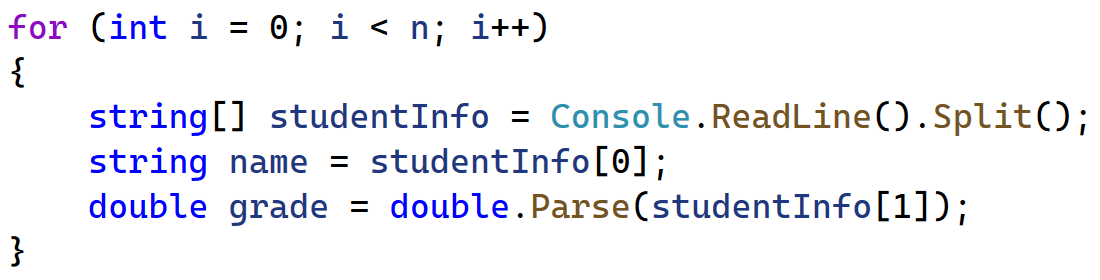
1. Прочетете броя на учениците **n**:



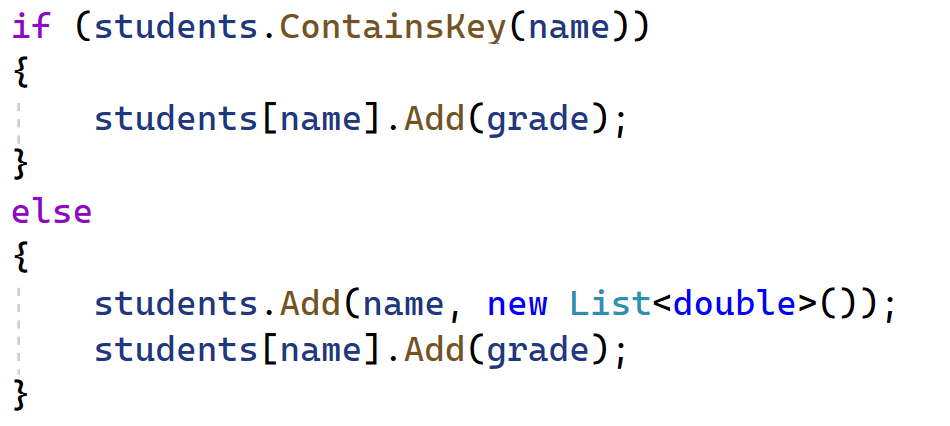
1. Създайте **речник** (string  List<double>), който да съхранява техните оценки:



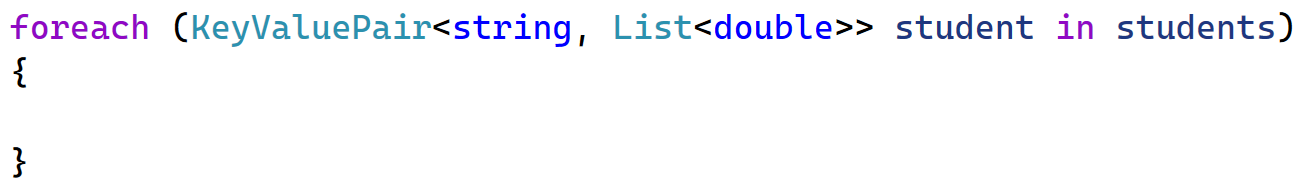
1. След това създайте **for-цикъл**, който итерира от 0 до **броя на учениците**. За всяка итерация на цикъла прочетете данните на ученика и ги разделете на две **променливи** - **име** и **оценка**:



1. Добавете проверка дали името на **ученикът** **присъства** като ключ в речника. Ако е така, добавете **неговата** **оценка** към вече съществуващия **списък**. В противен случай, трябва да **добавите** **ученика** и да сложите като стойност **нов списък** с **една оценка**:



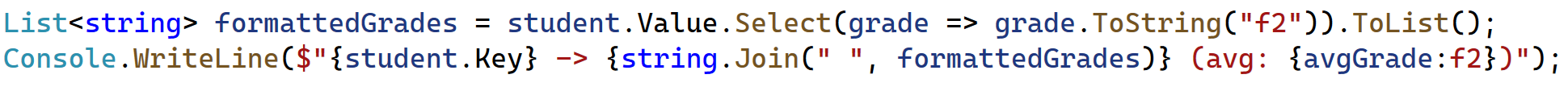
1. Създайте **foreach-цикъл**, който ще обхожда всички ученици:



1. В него създайте **променлива**, която ще **пресмята** средния успех на ученика чрез метода .Average():



1. Накрая **отпечатайте** изхода, като **форматирате** **оценките** и **средния успех** на ученика **до втората цифра след десетичната точка**:



## Хранителен магазин

Напишете програма, която **принтира и събира** информация за **хранителните магазини** в София. Докато не получите командата "Revision", ще полочвате команди във формата: "{магазин}, {продукт}, {цена}". Имайте предвид, че ако получите магазин, който съществува, просто добавете информацията за продукта.

Трябва да ги отпечатате подредени по име:

"{Хранителен магазин}->

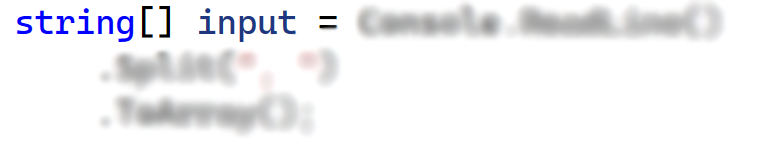
Product: {Продукт}, Price: {Цена}"

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| lidl, juice, 2.30  fantastico, apple, 1.20  kaufland, banana, 1.10  fantastico, grape, 2.20  Revision | fantastico->  Product: apple, Price: 1.2  Product: grape, Price: 2.2  kaufland->  Product: banana, Price: 1.1  lidl->  Product: juice, Price: 2.3 |
| tmarket, peanuts, 2.20  GoGrill, meatballs, 3.30  GoGrill, HotDog, 1.40  tmarket, sweets, 2.20  Revision | GoGrill->  Product: meatballs, Price: 3.3  Product: HotDog, Price: 1.4  tmarket->  Product: peanuts, Price: 2.2  Product: sweets, Price: 2.2 |

### Насоки

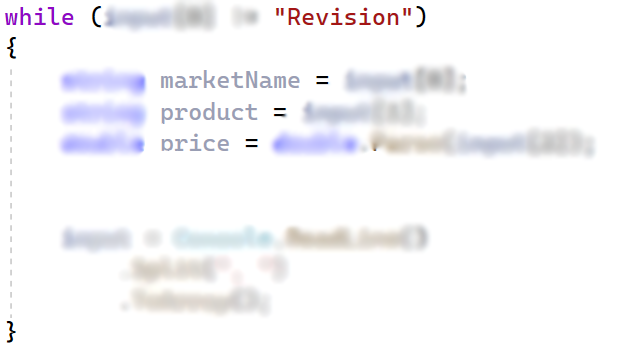
1. Първо прочете входа:



1. Създайте **сортиран** **речник**:



1. След това създайте **while-цикъл**, който ще обхожда докато нулевия елемент на масива **input** е различен от **"Revision"**. В него прочетете входа и го разделете на три **променливи** – **име**, **продукт** и **цена**:



1. Добавете информацията в **сортирания речник**.
2. Накрая отпечатайте **информацията**. Създайте **foreach-цикъл**, който да **обхожда** **всички** **магазини**. В него създайте още един **foreach-цикъл** за **обхождане** на **продуктите** на **магазина**:



## Градове по континент и държава

Напишете програма, която чете число **n**. След това ще получите **n-**на брой реда с **континент**, **държава** и **град**, разделени с **интервал**. Може да получите **няколко държави** в **един континент** и **няколко града** в **една държава**. Сложете ги в **сложен речник** и ги отпечатайте във формата, показан в примера.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 9  Europe Bulgaria Sofia  Asia China Beijing  Asia Japan Tokyo  Europe Poland Warsaw  Europe Germany Berlin  Europe Poland Poznan  Europe Bulgaria Plovdiv  Africa Nigeria Abuja  Asia China Shanghai | Europe:  Bulgaria -> Sofia, Plovdiv  Poland -> Warsaw, Poznan  Germany -> Berlin  Asia:  China -> Beijing, Shanghai  Japan -> Tokyo  Africa:  Nigeria -> Abuja |
| 3  Europe Germany Berlin  Europe Bulgaria Varna  Africa Egypt Cairo | Europe:  Germany -> Berlin  Bulgaria -> Varna  Africa:  Egypt -> Cairo |
| 8  Africa Somalia Mogadishu  Asia India Mumbai  Asia India Delhi  Europe France Paris  Asia India Nagpur  Europe Germany Hamburg  Europe Poland Gdansk  Europe Germany Danzig | Africa:  Somalia -> Mogadishu  Asia:  India -> Mumbai, Delhi, Nagpur  Europe:  France -> Paris  Germany -> Hamburg, Danzig  Poland -> Gdansk |

### Насоки

1. Използвайте **сложен речник** (string  (Dictionary  List<string>))
2. Проверете дали континента съществува в речника. Ако липсва, **добавете го**.
3. Проверете дали държавата съществува в речника. Ако липсва, **добавете я**.
4. Преминете през всички **ключ-стойност** двойки в речника и отпечатайте **резултата**.

## Уникални имена

Напишете програма, която чете **списък от имена** и отпечатва **уникалните** имена.

### Примери

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |
| 8  Ivan  Pesho  Ivan  Stamat  Pesho  Alice  Peter  Pesho | Ivan  Pesho  Stamat  Alice  Peter |  | 7  Lyle  Bruce  Alice  Easton  Shawn  Alice  Shawn  Peter | Lyle  Bruce  Alice  Easton  Shawn |  | 6  Roki  Roki  Roki  Roki  Roki | Roki |

### Насоки

1. Сложете имената в HashSet<string>, за да вземете уникалните имена.