# Лаб: Сетове и речници

Тествайте решението в Judge: <https://judge.softuni.bg/Contests/3178/Additional-Exercises>.

## Най-големите 3 числа

Прочетете списък от числа. Отпечатайте **трите най-големи** числа. Ако има **по-малко от 3 числа**, отпечатайте **всички**.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 10 30 15 20 50 5 | 50 30 20 |

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 20 30 | 30 20 |

### Насоки

1. Прочетете масив от числа
2. **Сортирайте масива** чрез LINQ

A picture containing text

Description automatically generated

1. Отпечатайте **първите 3 числа** чрез цикъл

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

Напишете програма, която чете **масив от числа** и отпечатва **колко пъти се срещат** във **възходящ ред.**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| -2 0 0 2 | -2 -> 1  0 -> 2  2 -> 1 |

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 8 2 2 8 2 | 2 -> 3  8 -> 2 |

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 1 5 1 3 | 1 -> 2  3 -> 1  5 -> |

### Насоки

1. Прочитане на масива от числа:

Graphical user interface, text

Description automatically generated

1. Използване на SortedDictionary<double, int> за броя на срещанията.



1. Минаваме през всяко едно число и увеличаваме бройката - counts[num]. Ако няма дадено число в речника, слагаме в речника стойност 1 - counts[число] = 1, защото даденото число се среща за първи път.

Graphical user interface, text, application

Description automatically generated

1. Минаваме през всяко число от **речника** и отпечатваме **колко пъти се** **среща**.

Text

Description automatically generated with low confidence

## Нечетен брой повторения

Напишете програма, която извлича **всички думи от низ**. Отпечатайте думита, които се повтарят нечетен брой пъти (няма да има значение дали буквите на думите ще бъдат малки или големи).

* Ще получите думите на **един ред**, разделени с **интервал**.
* Отпечатайте думите с **малки букви**. Трябва да бъдат сортирани по ред на задаване.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Java C# PHP PHP JAVA C java | java c# c |
| 3 5 5 hi pi HO Hi 5 ho 3 hi pi | 5 hi |
| a a A SQL xx a xx a A a XX c | a sql xx c |

### Насоки

1. Прочетете **думите** от конзолата, като ги разделите с **интервал**



1. Използвайте **речник** (низ à **число**) за да следите броя на думите



1. Минете през **всички елементи** в масива и започнете да ги **броите**

Graphical user interface, text, application

Description automatically generated

1. Минете през речника и отпечатайте думите, които се срещат **нечетен брой пъти**

Text

Description automatically generated

## Синоними

Напишете програма, която съдържа **речник за синоними**. **Ключът** на речникът е **думата**. Неговата **стойност** ще **бъде списък със синоними** на думата. Ще получите на първия ред числото **n** - **броя на думите**. След всяка дума ще получите нейния **синоним**, затова имайте предвид, че ще прочитате от конзолата **n\*2** реда. На редовете ще получите дума и синоним във формата:

* {**дума**}
* {**синоним**}

Ако получите дума, която **вече я има**, просто **добавете нейния синоним в списъка**.

Отпечатайте думите във формата:

**{дума} - {синоним1, синоним2… синонимN}**

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 3  cute  adorable  cute  charming  smart  clever | cute - adorable, charming  smart - clever |
| 2  task  problem  task  assignment | task – problem, assignment |

### Насоки

1. Използвайте **речник (string -> List<string>), за да складирате синонимите.**

****

1. **Прочете n\*2 реда и на всеки ред прочетете нова дума**
2. **Добавете думата, ако я няма в речника**

Text

Description automatically generated with low confidence

1. **Добавете синонима като стойност в думата**

****

1. **Отпечатайте всички думи със синонимите им, както е показано в примерите по-горе**

## Паркинг

Напишете програма, която:

* **Добавя** номера на колата всеки път, когато колата **влезе** в паркинг
* **Изтрива** номера на колата всеки път, когато колата **излезе** от паркинга

Входът ще бъде низ във формата: "**IN/OUT, номер на колата"**. Ще получавате команди, докато не получите **"END"**.

Накрая отпечатайте, **кои коли са останали** в паркинга

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| IN, CA2844AA  IN, CA1234TA  OUT, CA2844AA  IN, CA9999TT  IN, CA2866HI  OUT, CA1234TA  IN, CA2844AA  OUT, CA2866HI  IN, CA9876HH  IN, CA2822UU  END | CA9999TT  CA2844AA  CA9876HH  CA2822UU |
| IN, CA2844AA  IN, CA1234TA  OUT, CA2844AA  OUT, CA1234TA  END | Parking Lot is Empty |

### Бележки

* Номерата винаги ще бъдат **уникални**

## Партито на СофтУни

СофтУни има парти. Много хора са поканени, затова ги делим на **VIP** и **нормални**. Когато гост пристигне, проверяваме дали е в един от двата списъка.

Имената на всички гости ще имат дължина от **8 символа**.

Всички **VIP** гости ще започват с **цифра**.

Първо ще получите **резервациите на гостите**. Ще имате **2 възможни команди**:

* "PARTY" – след тази команда ще получите **гостите**, които **наистина ще присъстват** на партито
* "END" – когато получите тази команда, трябва **програмата да спре** и да отпечатате **изхода**

Накрая трябва да отпечатате броя на хората, които **не са присъствали**. След това отпечатайте техните номера. **VIP гостите са с предимство**.

### Примери

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Вход** | **Изход** |
| 7IK9Yo0h  9NoBUajQ  Ce8vwPmE  SVQXQCbc  tSzE5t0p  PARTY  9NoBUajQ  Ce8vwPmE  SVQXQCbc  END | 2  7IK9Yo0h  tSzE5t0p | m8rfQBvl  fc1oZCE0  UgffRkOn  7ugX7bm0  9CQBGUeJ  2FQZT3uC  dziNz78I  mdSGyQCJ  LjcVpmDL  fPXNHpm1  HTTbwRmM  B5yTkMQi  8N0FThqG  xys2FYzn  MDzcM9ZK  PARTY  2FQZT3uC  dziNz78I  mdSGyQCJ  LjcVpmDL  fPXNHpm1  HTTbwRmM  B5yTkMQi  8N0FThqG  m8rfQBvl  fc1oZCE0  UgffRkOn  7ugX7bm0  9CQBGUeJ  END | 2  xys2FYzn  MDzcM9ZK |

## Брой букви в низ

Напишете програма, която чете **текст** от конзолата. Вашата задача е да преброите **колко пъти се среща една буква**. Отпечатайте резултата в **азбучен** (лексикографски) ред.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| SoftUni rocks | : 1 time/s  S: 1 time/s  U: 1 time/s  c: 1 time/s  f: 1 time/s  i: 1 time/s  k: 1 time/s  n: 1 time/s  o: 2 time/s  r: 1 time/s  s: 1 time/s  t: 1 time/s |
| Did you know Math.Round rounds to the nearest even integer? | : 9 time/s  .: 1 time/s  ?: 1 time/s  D: 1 time/s  M: 1 time/s  R: 1 time/s  a: 2 time/s  d: 3 time/s  e: 7 time/s  g: 1 time/s  h: 2 time/s  i: 2 time/s  k: 1 time/s  n: 6 time/s  o: 5 time/s  r: 3 time/s  s: 2 time/s  t: 5 time/s  u: 3 time/s  v: 1 time/s  w: 1 time/s  y: 1 time/s |

## \*Класиране

Напишете програма, която класира кандидат-стажантите по техния **брой точки** от задачата за интервю и техния изпит в СофтУни. Ще получите редове във формата: **"{Изпит}:{парола за изпита}",** докато не получите **"end of contests"**. Трябва да запазите резултата, защото ще ви бъде нужен за по-късно. След това ще получите **още един вход** във формата **"{Изпит}=>{Парола за изпита}=>{Потребителско име}=>{точки}"**, докато не получите **командата** **"end of submissions"**. Ето какво трябва да знаете:

* Проверете дали **изпита** **е** **валиден**
* Проверете дали **паролата за изпита е валидна**
* Запомнете изпита на потребителя с неговото участие (един потребител може да има много изпити) и точките, които е изкарал на изпита. Ако получи изпит, на който **вече е участвал**, **обновете** неговите **точки**, ако има **повече**.

Накрая отпечатайте информацията за потребителя **с най-много точки** във формата:

**"Best candidate is {потребителско име} with total {точки} points."**. След това отпечатайте всички студенти, **подредени по техните имена**. За всеки потребител отпечатайте неговите участия и точки. Сортирайте ги в **низходящ ред** по точки във формата:

**"{Потребителско име1}**

**# {изпит1} -> {точки}**

**# {изпит2} -> {точки}**

**{Потребителско име2}**

**…"**

### Вход

* Ще получите низ във формата, описан по-горе, докато не получите командите, както е описано.

### Изход

* На **първия ред** отпечатайте **най-добрия потребител** във формата по-горе.
* На **следващите редове** трябва да отпечатате студентите, подредени във формата по-горе.

### Ограничения

* **Няма** да има **два** **еднакви изпита**.
* Низът ще бъде от **ASCII** символи освен **(:, =, >).**
* Числата ще бъдат в обхвата **[0 - 10000].**
* **Втория** вход винаги ще бъде **валиден**.
* Няма да има **двама или повече** потребители с **еднакви точки.**

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Part One Interview:success  Js Fundamentals:JSFundPass  C# Fundamentals:fundPass  Algorithms:fun  end of contests  C# Fundamentals=>fundPass=>Tanya=>350  Algorithms=>fun=>Tanya=>380  Part One Interview=>success=>Nikola=>120  Java Basics Exam=>JSFundPass=>Parker=>400  Part One Interview=>success=>Tanya=>220  OOP Advanced=>password123=>BaiIvan=>231  C# Fundamentals=>fundPass=>Tanya=>250  C# Fundamentals=>fundPass=>Nikola=>200  Js Fundamentals=>JSFundPass=>Tanya=>400  end of submissions | Best candidate is Tanya with total 1350 points.  Ranking:  Nikola  # C# Fundamentals -> 200  # Part One Interview -> 120  Tanya  # Js Fundamentals -> 400  # Algorithms -> 380  # C# Fundamentals -> 350  # Part One Interview -> 220 |
| Java Advanced:funpass  Part Two Interview:success  Math Concept:asdasd  Java Web Basics:forrF  end of contests  Math Concept=>ispass=>Monika=>290  Java Advanced=>funpass=>Simon=>400  Part Two Interview=>success=>Drago=>120  Java Advanced=>funpass=>Petyr=>90  Java Web Basics=>forrF=>Simon=>280  Part Two Interview=>success=>Petyr=>0  Math Concept=>asdasd=>Drago=>250  Part Two Interview=>success=>Simon=>200  end of submissions | Best candidate is Simon with total 880 points.  Ranking:  Drago  # Math Concept -> 250  # Part Two Interview -> 120  Petyr  # Java Advanced -> 90  # Part Two Interview -> 0  Simon  # Java Advanced -> 400  # Java Web Basics -> 280  # Part Two Interview -> 200 |