# Упражнение: Списъци

Тествайте решението в Judge: <https://judge.softuni.org/Contests/4150/02-Lists-Basics>.

## Трикът на Гаус

Напишете програма, която **сумира** всички **числа** в **списък** в следният формат:

*първо + последно, първо + 1 + последно - 1, първо + 2 + последно - 2, … първо + n, последно - n.*

Graphical user interface, text

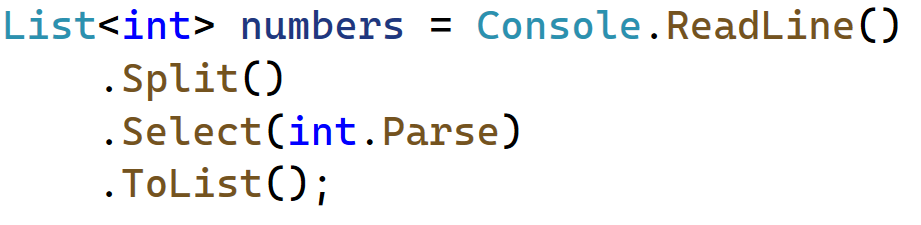
Description automatically generated with medium confidence

### Примери

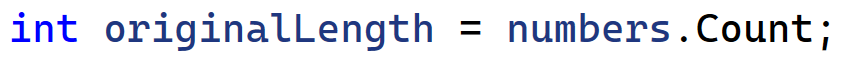
|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 1 2 3 4 5 | 6 6 3 |
| 1 2 3 4 | 5 5 |

### Насоки

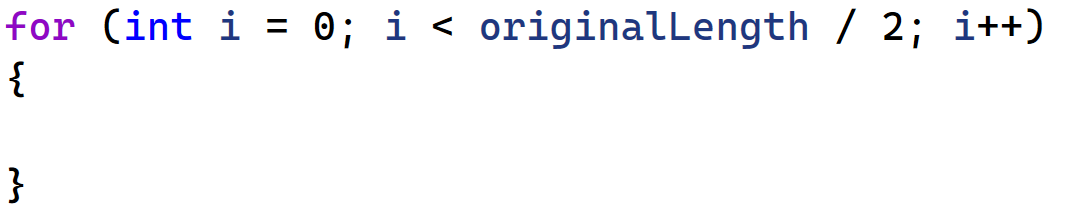
1. Прочете **списъка** от **числа**:



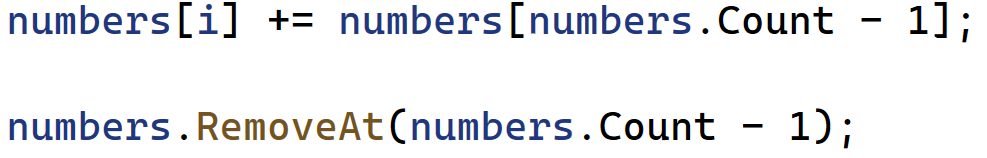
1. Създайте **променлива**, която ще има стойност **оригиналната дължина на списъка**:



1. Създайте for**-цикъл**, който ще итерира докато неговия брояч е **по-малък от дължината** на **списъка /2**:



1. На всяка итерация **добавете** към **първия** **елемент** на списъка стойността на **последния елемент** и след това **премахнете последния елемент**, за да **запазите правилния брой елементи**:



1. Накрая отпечатайте **списъка**:



## Еднакви числа

Напишете програма, която да събира всички съседни числа с еднаква сума в списъкът отляво надясно.

* След като **сумирате** **двете числа** , **резултатът** може да бъде равен на **съдеда** на второто число и трябва да се **сумира също** (както е показоно по-долу).
* Винаги сумирайте **най-левите два равни съседа** (ако има няколко равни двойки съседи)

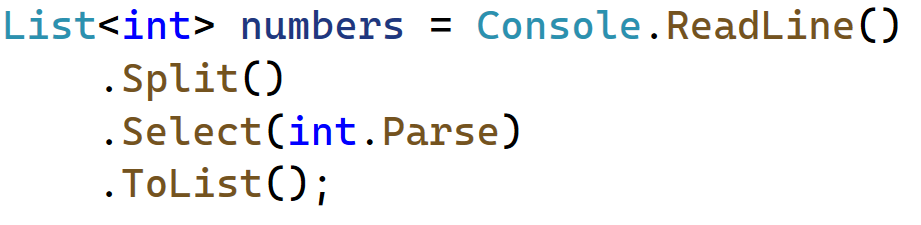
Програмата свършва, когато няма повече съседни числа с еднакви стойности. Накрая отпечатайте крайният резултат.

### Примери

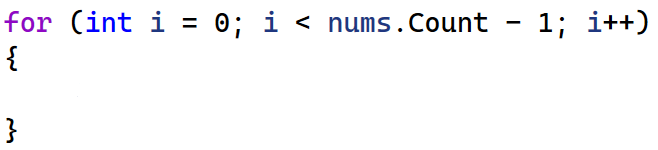
|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 3 3 6 1 | 12 1 | **3 3** 6 1  **6 6** 1  12 1 |
| 8 2 2 4 8 16 | 16 8 16 | 8 **2 2** 4 8 16  8 **4 4** 8 16  **8 8** 8 16  16 8 16 |
| 5 4 2 1 1 4 | 5 8 4 | 5 4 2 **1 1** 4  5 4 **2 2** 4  5 **4 4** 4  5 8 4 |

### Насоки

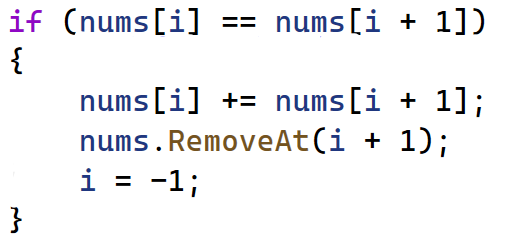
1. Прочетете **списъка от числа**.



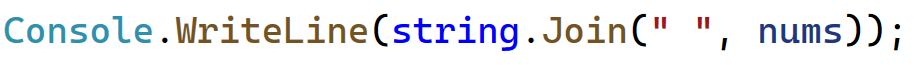
1. Създайте **for**-цикъл, който ще итерира до дължината на списъка **-1**:



1. Проверете дали числото на **сегашния индекс** и **следващото** са **еднакви**. Ако е така, нулирайте for**-цикъла**:



1. Накрая отпечатваме всички числа **разделени с интервал**:



## Манипулация на списък

Напишете програма, която чете **списък от числа**. Докато не получите **"end",** ще получавате следните възможни **команди**:

**Add {число}:** Добавяне на число в края на списъка.  
**Remove {число}:** Премахване на число от списъка.  
**RemoveAt {индекс}:** Премахване на число с определен индекс.  
**Insert {число} {индекс}:** Вмъкване на число на определен индекс.

**Бележка: Не всички команди ще бъдат валидни!**

Когато получите **"****end",** принтирайте **финалния** списък (**числата трябва да бъдат разделени с интервал**).

### Примери

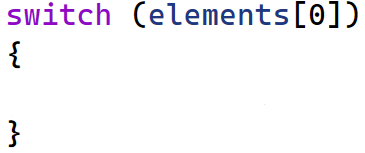
|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 4 19 2 53 6 43  Add 3  Remove 2  RemoveAt 1  Insert 8 3  end | 4 53 6 8 43 3 |

### Насоки

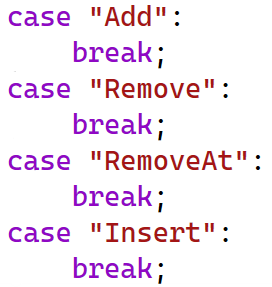
1. **Прочитете** списъка.
2. Създайте **променлива** за **четене** на **входа.**
3. След това създайте while**-цикъл**.
4. В него прочетете входа.
5. Ако променливата за входа има стойност "**end**", излезте от цикъла.
6. Превърнете входа в **масив**:



1. Създайте **switch** конструция за различните команди:



1. Имплементирайте всички **команди**:



1. Накрая **отпечатайте числата**, разделени с **интервал**.

## Обединяване на списъци

Напишете програма, която получава два списъка с числа. Крайният списък трябва да съдържа числа от двата списъка. **Първият елемент** на крайният списък е **от първия елемент на първия списък** и **вторият елемент** **е първият елемент от вторият списък** и така нататък. Ако дължината на двата списъка е **различна**, просто **добавете оставащите елеменити** в **края** на списъка.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| **3 5 2 43 12 3 54 10 23**  **76 5 34 2 4 12** | **3** **76** **5** **5** **2** **34** **43** **2** **12** **4** **3** **12** **54** **10** **23** |
| **76 5 34 2 4 12**  **3 5 2 43 12 3 54 10 23** | **76** **3** 5 **5** **34** **2** **2** **43** **4** **12** **12** **3** **54 10 23** |

### Насоки

1. Четем **два списъка**.
2. Създаваме **краен списък**.
3. Стартираме **цикъл**, който минава през всички елементи и спира до **дължината на по-малкия списък**.
4. Накрая **добавяме оставащите елементи** (ако има някакви) в края на списъка.

## Списък от продукти

Прочетете числото **n**, след което **n** на брой редове с продукти. Принтирайте **номериран списък** с продуктите, **подредени по азбучен ред**.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 4  Potatoes  Tomatoes  Onions  Apples | 1.Apples  2.Onions  3.Potatoes  4.Tomatoes |

### Насоки

1. Първо прочетете числото n.
2. След това създайте **списък от низове** за продуктите.
3. Създайте for**-цикъл** и прочетете продуктите.
4. В него добавете **продуктите** към списъка.
5. След цикъла **сортирайте продуктите** по **азбучен ред**.
6. Накрая **принтираме списъка** чрез for**-цикъл**.

## Премахнете отрицателните числа обърнете списъка

Прочетете **списък от числа**. **Премахнете всички отрицателни** числа и принтирайте останалите числа в **обратен ред**. В случай, в които няма числа отпечатайте "empty".

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 10 -5 7 9 -33 50 | 50 9 7 10 |
| 7 -2 -10 1 | 1 7 |
| -1 -2 -3 | empty |

## Влак

Напишете задача, в която на първият ред получавате **списък от информация за вагони** (числа). Всяко число представя **броя на пасажерите във всеки вагон**. На нов ред ще получите **капацитета на вагоните**. На следващите редове, докато командата е различна от **"end"**, ще получавате два типа команди:

* **Add {пасажери}** – **добавате вагон** със съотвения **брой пасажери** в него.
* **{пасажери}** - намирете вагон, в който **можете да добавите пасажерите**, като стартирате търсенето от първивят вагон.

Накрая **отпечатайте броя на хората** във всички вагони (вагоните трябва да бъдат разделени с **интервал**).

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 32 54 21 12 4 0 23  75  Add 10  Add 0  30  10  75  end | 72 54 21 12 4 75 23 10 0 |
| 0 0 0 10 2 4  10  Add 10  10  10  10  8  6  end | 10 10 10 10 10 10 10 |