# Упражнение: Списъци

Тествайте решението в Judge: <https://judge.softuni.bg/Contests/3170/Lists>.

## Влак

Напишете задача, в която на първият ред получавате **списък от 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 входа:

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

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

**Примерен вход**

|  |  |
| --- | --- |
| **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 |

## Списък с промени

Напишете програма, която да чете **списък** от **числа** от **конзолата** и **команди**, с който да се **маниполира** списъкът. Вашата програма приема следните команди:

* **Delete {елемент}** – изтрийте всички числа в списъкът, които са равни даденият елемент.
* **Insert {елемент} {позиция}** – добавете елемент на определена позиция.

Програмата спира, когато се въведе командата **"end"**. Принтирайте всички числа разделени с **интервали.**

**Примерен вход**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 1 2 3 4 5 5 5 6  Delete 5  Insert 10 1  Delete 5  end | 1 10 2 3 4 6 |
| 20 12 4 319 21 31234 2 41 23 4  Insert 50 2  Insert 50 5  Delete 4  end | 20 12 50 319 50 21 31234 2 41 23 |

## Операции на списък

Напишете програма, която на първият ред прочита **списък** от **числа**. На следващите редове ще получавате команди, докато не се въведе **"End"**. Възможните **команди** са:

* **Add {число} – добавя число в края на списъка.**
* **Insert {число} {индекс} – добавя число на определен индекс.**
* **Remove {индекс} – премахва число от определен индекс.**
* **Shift left {брой} – премества първото число на края на списъка определен 'брой' пъти.**
* **Shift right {пъти} – премества последното число в началото на списъка определен 'брой' пъти.**

**Бележка: не винаги индексите ще бъдат в границите на списъка. В такъв случай отпичатайте "Invalid index"**

**Примерен вход**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 1 23 29 18 43 21 20  Add 5  Remove 5  Shift left 3  Shift left 1  End | 43 20 5 1 23 29 18 |
| 5 12 42 95 32 1  Insert 3 0  Remove 10  Insert 8 6  Shift right 1  Shift left 2  End | Invalid index  5 12 42 95 32 8 1 3 |

## Номер на бомбите

Напишете програма, която **чете редица от числа** и **намира специални бомби** с номер и определена **мощ**. Вашата задача е да **взревите всички бомби** и според силата им -**техните съседи отляво и отдясно**. Детонациите се извършват **отляво надясно** и всички детонирани номера **изчезват**. Отпечатайте **сумата на оставащите елементи** от редицата.

### Примерен вход и пример

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 1 2 2 4 2 2 2 9  4 2 | 12 | Специалното число е **4** със сила **2**. След детонацията нашата редица [1, 2, 9] е със сума 12. |
| 1 4 4 2 8 9 1  9 3 | 5 | Специалното число е **9** със сила **3**. След детонацията нашата редица [1, 4] е със сума 5. Тъй като числото 9 има само един съсед отдясно ние махаме само него (едно число вместо 3) |
| 1 7 7 1 2 3  7 1 | 6 | Детонацията се изпълнява отляво надясно. Ние не може да детонираме втората **7**, защото тя вече е взривена от първата. Оцелелете числа са [1, 2, 3]. Тяхната сума е 6. |
| 1 1 2 1 1 1 2 1 1 1  2 1 | 4 | Червените и жълтите числа изчезват в две последователни редици. Резултатът на редицата е [1, 1, 1, 1]. Сумата e 4. |

## Игра на карти

Ще ви бъдат дадени 2 тестета от карти. Нека да приемим, че има двама играчи. Вие трябва да намерите печелившото тесте и победителя.

Започваме от началото на двете тестета. Сравняваме картите от първото и второто тесте. Играчът с печелившата карта печели и прибира картите от ръката в своето тесте (първо се добавя картата на печелившия и след това се прибавя спечелената карта най-долу в тестето). Другият играч трябва да **я премахне** от тестето си. Ако и двете карти имат еднаква стойност - никой не печели и двете **карти** се **премахват от тестета им**. Играта свършва, когато някой от играчите остане без карти. Трябва да отпечатате победителя на конзолата и неговите останали карти: **"{First/Second} player wins! Sum: {сума}".**

**Примерен вход**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 20 30 40 50  10 20 30 40 | First player wins! Sum: 240 |
| 10 20 30 40 50  50 40 30 30 10 | Second player wins! Sum: 50 |

## \*Заплахата от анонимните

*Анонимните са създали кибер вирус, който краде данните на ЦРУ. Като водещ security developer в ЦРУ, вие трябва да анализирате софтуера на вируса и да наблюдавате действията му върху данните. Вирусът е известен със своята иновативна и невероятно умна техника за обединяване и разделяне на данни.*

Ще получете **един ред**, който съдържа **низове** разделени с **интервал**. Низовете могата да съдъражат всякакъв **ASCII символ** освен интервал. След това ще получавате команди в следният формат:

* merge {стартов индекс} {краен индекс}
* divide {индекс} {дял}

Всеки път когато получавате **команда за обединяване**, вие трябва да обедините всички елементи от **стартовия индекс** до **последния индекс**. С други думи трябва да ги свържите.

**Пример**: {abc, def, ghi} -> merge 0 1 -> {abcdef, ghi}

Ако има даден индекс, който е **извън списъка**, вие трябва да обедените само елементите **в дължината на списъка**.

Всеки път, когато получите **команда за разделяне**, трябва да **разделите елемента** на дадения индекс на **няколко по-малки подниза** с **еднаква дължина**. **Броят** на **поднизовете** трябва да е **равен** на **даденият дял**.

**Пример**: {abcdef, ghi, jkl} -> divide 0 3 -> {ab, cd, ef, ghi, jkl}

Ако низът **не може** да се **раздели точно** на **даденият дял**, **направете всички дялове** с **изключение** на **последния с еднаква дължина**.

Пример: {abcd, efgh, ijkl} -> divide 0 3 -> {a, b, cd, efgh, ijkl}

Четенето на команди спира, когато се въведе командата "3:1". След тази команда трябва да отпичатате всички елементи разделени с интервал.

### Вход

* На **първият ред** се чете списък от данни
* На **следващите редове** ще **получите команди** в посочения **по-горе формат**.
* Четенето на команди спира, когато се въведе командата "3:1"

**Изход**

* На изхода отпечатайте на един ред елементите на списъка **разделени с интервал**.

**Бележки**

* **Низовете** в **списъка** могата да съдъражат всякакъв **ASCII символ** освен **интервал**.
* **Стартовият** и **крайният индекс** ще бъдат в обхвата **[-1000, 1000]**.
* **Последният индекс** винага ще бъде **по-голям** от **стартовият индеск**.
* **Индексът** в командата за **разделяне** **винаги** ще бъде **вътре** в списъка.
* **Дялът** ще бъде в **обхвата [0, 100]**.
* Маскимално **време/памет:** **100ms / 16MB**

**Примерен вход**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Ivo Johny Tony Bony Mony  merge 0 3  merge 3 4  merge 0 3  3:1 | IvoJohnyTonyBonyMony |
| abcd efgh ijkl mnop qrst uvwx yz  merge 4 10  divide 4 5  3:1 | abcd efgh ijkl mnop qr st uv wx yz |

## \*Pokemon Don’t Go

*Ели обича да играе Pokemon Go, но играта фалира. Тогагва създателите изпаднали в депресия създадоха Pokemon Don’t Go и Ели я играе. В Pokemon Don’t Go близките покемони се отдаличават, а тези който са далeче се приближават.*

Ще Ви бъде дадена **редица** от **числа** разделена от **интервал**-дистаницията на покемоните. След това ще започнете да **получавате цели числа**, които ще **съответстват** на **индекси** в **редицата**.

Когато **получавате индекс**, Вие трябва да **премахнете елемент** на **този индекс** от **редицата**(Все едно сте хванали покемон)

* Трябва да **увеличите** **стойността** на **всички елементи** в **редицата**, които са **по-малки или равни** на **премахнатия елемент**, със **стойността** на **премахнатия елемент**.
* Трябва да **намалите стойността** на **всички елементи** в редицата, които са **по-големи** от **премахнатия елемент**, със **стойността** на **премахнатия елемент**.

Ако **даденият индекс** е **по-малък** от **0,** **премахнете първия елемент от редицата** и **копирайте последният елемент** на негово място.

Ако **дадетният индекс** е **по-голям** от **дължината** на **редицата, премахнете последният елемент** и **копирайте първия елемент** на негово място.

Програмата **приключва**, когато **редицата няма елементи** (няма останали покемони, които Ели да хване).

### Вход

* На **първият ред** се чете **списъкът** от **числа**, **разделени** по **интервал.**
* На **следващите редове** ще получавате **числа**-**индиксите**.

**Изход**

* Когато програмата приключи, принтираме **сумата** на **премахнатите елементи**

**Бележки**

* Входните данни ще се състоят **само** от **числа** в **обхвата [-2.147.483.648, 2.147.483.647]**.

### Примерен вход и пример

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 4 5 3  1  1  0 | 14 | **Редицата** ни е **{4, 5, 3}**. Индексът е **1**.  **Премахваме 5** и **увеличаваме всички по-ниски** и **намаляме всички по-големи**.  В този случай няма **по-високи** от **5**.  Резултатът е **{9, 8}**.  **Индексът** е **1**. Премахваме **8** и **намаляваме всички по-големи** от него.  Резултатът е **{1}**.  **Индексът** е **0**. Премахваме **1**.  Принтираме **сумата** на **премахнатите елементи**, защото **няма повече числа** в колекцията.  5 + 8 + 1 = 14. |
| 5 10 6 3 5  2  4  1  1  3  0  0 | 51 | **Стъпка 1**: {11, 4, 9, 11}  **Стъпка 2**: {22, 15, 20, 22}  **Стъпка 3**: {7, 5, 7}  **Стъпка 4**: {2, 2}  **Стъпка 5**: {4, 4}  **Стъпка 6**: {8}  **Стъпка 7**: {} **(празно).**  **Резултат** = 6 + 11 + 15 + 5 + 2 + 4 + 8 = 51. |

## \*SoftUni Course Planning

Вашата задача е да помогнете за планирането на курса "Основни на Технологията" като проследявате лекциите и съответно упражненията към тях.

На първият ред получите **първоначалния график за лекциите и упражненията**, разделени с **запетая и интервал ", "** за следващият курс. Но преди да стартира курса може да се направят, някой промени. Докато не получите командата **"course start",** вие ще получавате **команди за модефициране**. Възможните команди са:

* **Add:{лекция} – добавяне на лекция в края, ако не съществува.**
* **Insert:{лекция}:{индекс} – добавяне лекция на даден индекс, ако не съществува.**
* **Remove:{лекция} – премахване на лекция, ако съществува.**
* **Swap:{лекция}:{лекция} – смяна на местата на два лекций, ако съществуват.**
* **Exercise:{лекция} - добавяне на упражнение в графика веднага след индекса на лекцията. Ако лекцията съществува и няма упражнение добавете упражнението в следния формат „{Лекция}- Exercise “. Ако лекцията не съществува, добавете я в края на графика на курса, последвана от упражнението.**

**Всеки път, когато разменяте или премахвате лекция, трябва да направите същото с упражненията, ако има такива към нея.**

### Вход

* първи ред – първоначалния график на уроците – низове, разделени със запетая и интервал **", "**
* до **"начало на курса"** ще получавате команди във формата, описана по-горе

**Изход**

* Отпечатайте целия график на курса, като всяка лекция трябва да бъде на нов ред с нейния номер (индекс) в графика: "{индекс на лекцията}.{Име на лекцията}"
* Маскимално **време/памет:** **100ms / 16MB**

### Примерен вход и пример

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| Data Types, Objects, Lists  Add:Databases  Insert:Arrays:0  Remove:Lists  course start | 1.Arrays  2.Data Types  3.Objects  4.Databases | Получаваме първоначалния график.  След това добавяме лекция за база данни, защото тя не съществува.  Вмъкваме в дадения индекс лекция за масиви, защото тя не съществува.  След получаване на последната команда премахваме списъци и отпечатваме целия график. |
| **Вход** | **Изход** | **Обяснения** |
| Arrays, Lists, Methods  Swap:Arrays:Methods  Exercise:Databases  Swap:Lists:Databases  Insert:Arrays:0  course start | 1.Methods  2.Databases  3.Databases-Exercise  4.Arrays  5.Lists | Разменяме двете лекции, защото и двете съществуват.  След като получим командата Exercise, виждаме, че такава лекция не съществува и затова добавяме лекцията в края, последвана от упражнението.  Разменяме лекциите по списъци и база данни,  База данни-Exercise също се премества след лекция Бази данни.  Пропускаме следващата команда, защото вече има такава лекция в графика ни. |