# Упражнение: Списъци

Тествайте решението в Judge: <https://judge.softuni.org/Contests/4151/02-Lists-Advanced>.

## Манипулация на списъци (Advanced)

В тази задача ще трябва да имплентираме **по-сложни команди** на списъка (**разширяване на предходната задача**). Отново ще четете списък и команди, докато не получите **"end":**

* **Contains {число}** – проверете дали число съществува в списъка. Ако е така принтирайте **"Yes**", в противен случай **"No such number".**
* **PrintEven** – принтирайте **всички четни числа разделени с интервал.**
* **PrintOdd** – принтирайте **всички нечетни числа разделени с интервал.**
* **GetSum** – принтирайте **сумата на всички числа**.
* **Filter {условие} {число}** – отпечатайте всички числа, които отговарят на **даденото условие**. Условията могат да бъдат **"<"**, "**>"**, **">="**, **"<="**.

След командата **"end"** принтирайте списъка, ако има някакви промени от оригиналния списък. **Промените** се правят само от командите от **предходната задача**.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 2 13 43 876 342 23 543  Contains 100  Contains 543  PrintEven  PrintOdd  GetSum  Filter >= 43  Filter < 100  end | No such number  Yes  2 876 342  13 43 23 543  1842  43 876 342 543  2 13 43 23 |

### Насоки

1. Прочетете **списъка**.
2. Създайте **променлива** от **тип** bool, чрез която ще проверявате дали **списъка** е **празен**.
3. Създайте **променлива**, чрез която ще **четете входните данни**.
4. Създайте while-цикъл, който ще итерира докато **променливата** за **входните данни** е различна от **"end"**.
5. В него създайте switch конструкция или поредица от if проверки за изпълнението на **всяка команда**:
   1. Contains {**число**} - проверете дали число съществува в списъка чрез **метода** Contains().
   2. GetSum – обходете **всички числа** чрез for-цикъл или използвайте **метода** Sum().
   3. Filter {**условие**} {**число**} - създайте switch case за всяко условие - **'<', '>', ">="** или **"<="**.
6. Принтирайте **списъка**, ако има някакви промени от **оригиналния списък**.

## Съобщения

Ще ви бъде даден **списък от числа** и **низ**. За всеки елемент трябва да **съберете неговите цифри**, за да получите определен индекс от **низа**. Ако индексът е **по-голям от дължината**, започнете да броите от самото начало (така винаги ще има валиден индекс). След като **вземете елемента**, го премахнете от низа. Така за следващият индекс, низа ще бъде с един символ по-малко.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 9992 562 8933  This is some message for you | hey |

### Насоки

1. Прочетете **списъка от числа**.
2. Прочетете **низа**.
3. Създайте for**-цикъл**, за да обходите **всеки елемент** в **списъка**.
4. За **всяко число**, съберете **цифрите** му и ги **използвайте**, за да извлечете **символ от низа**.
5. **Променете низа**, като **премахнете използвания символ**.
6. Отпечатайте **низа**.

## Състезание

Напишете програма, която ще пресмята **коя кола ще победи в състезанието за коли**. Ще получите **списък от числа**. Всеки елемент представлява **необходимото време за всяка стъпка** (индекс). Ще има **две коли**. **Едната започва от лявата част**, а другото **от дясната част**. Средата на масива ще бъде **финалната линия**. Броят на елементите винаги ще бъде **четен**. Пресметнете **необходимото време** на всеки състезател да премине **финалната лента**. Отпечатайте **времето на победителя**. Ако имате **0** в списъка, трябва да намалите текущото време **с 20%**. Отпечатайте резултата в следния формат : **"The winner is {left/right} with total time: {общо време}".**

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 29 13 9 0 13 0 21 0 14 82 12 | The winner is left with total time: 53.8 |
| **Обяснения** | |
| Времето на левият състезател е (29 + 13 + 9) \* 0.8 (заради нулата) + 13 = 53.8  Времето на десният състезател е (82 + 12 + 14) \* 0.8 + 21 = 107.4  Отпечатваме левия състезател, защото има по-малко време | |

### Насоки

1. Прочетете **списъка от числа**.
2. Разделете **го** на **две части** - **лява** и **дясна** част от **финалната лента**.
3. Пресметнете времето **за всяка кола**, като съберете времената за всяка стъпка.
4. Ако има **0** в **списъка**, намалете **текущото време с 20%**.
5. Сравнете **времената** на **двете коли** и отпечатайте по-бързата кола и нейното време.

## Списък с промени

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

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

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

### Примери

|  |  |
| --- | --- |
| **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 първия 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:** **невинаги** индексите ще бъдат в **границите на списъка**. В такъв случай **отпечатайте** **"Invalid index"**

### Примери

|  |  |
| --- | --- |
| **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 **взривите 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.

### Примери

|  |  |  |
| --- | --- | --- |
| **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. |

## Скрити съобщения

Напишете програма, която чете **низ** и **открива** скритите съобщения. Трябва да приложете следният алгоритъм:

Нека нашият низ е “skipTest\_String044170”.

Запазваме цифрите от низа. След това ще премахнем всички числа от низа. В тази операция трябва да имате два списъка: списък с числа и списък без числа:

* Списък с числа: [0, 4, 4, 1, 7, 0]
* Списък без числа: [s, k, i, p, T, e, s, t, \_, S, t, r, i, n, g]

След това разделете списъка с числа на взети (на четни позиции) и пропуснати (на нечетни позиции):

* Списък от пропуснатите: [0, 4, 7]
* Списък от взетите: [4, 1, 0]

След това **минаваме** през двата списъка. **Пропускаме** символи{**индекс от списъка на пропуснатите числа**} от списъка без числа и добавяме символи {**индекс от списъка на взетите числа**} в крайният низ. Процедурата трябва да изглежда последният начин:

1. Взимаме **0** символа  Вземаме: "", пропускаме **4** символа (общо **0**)    
   Пропуснато: "**skipTest\_String**" Резултат: ""
2. Взимаме **4** символа Вземаме: "**Test"**, пропускаме **1** символ (общо **4**)  Пропуснато: "**skip**"  Резултат: "**Test**"
3. Взимаме **7** символа Взимаме: "**String**", пропускаме **0** символа (общо **9**) Пропуснато: ""  Резултат: "**TestString**"

Накрая отпечатваме крайния низ на конзолата.

### Вход

* На първия ред получавате **криптираното** съобщение като **низ**

### Изход

* На първия ред принтираме **декриптираното** съобщение като **низ**

### Бележки

* Броя на числата **винаги** ще бъде валиден.
* Криптираното съобщение винаги ще бъде от ASCII символи.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| T2exs15ti23ng1\_3cT1h3e0\_Roppe | TestingTheRope |
| O{1ne1T2021wf312o13Th111xreve!!@! | OneTwoThree!!! |
| this forbidden mess of an age rating 0127504740 | hidden message |