# Упражнения: Стек и опашка

Тествайте решението в Judge: <https://judge.softuni.bg/Contests/3172/Stack-Queue>.

## Обърнат низ

Напишете програма, която:

* **Прочита** низ
* **Обръщаго** като **използва** Stack<T>
* **Отпечатайте** резултата

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| I Love C# | #C evoL I |
| Stacks and Queues | seueuQ dna skcatS |

### Насоки

* Използвайте Stack<string>
* Използвайте мотодите **Push()**, **Pop()**

## Сума от стек

Напишете програма, която:

* **Чете** вход от числа и ги добавя в стек
* **Чете команди** докато не получите **"end"**
* **Отпечатва сумата** на останлите елементи от **стека**

### Вход

* **На първия** ред прочитете стека **от числа**
* На **следващите редове** ще получавате команди- **низ** с **едно** или **две** числа, докато не получите **"end"**
* Ако **командата** е **"add",** ще получите **две числа** след командата, които трябва да **добавите** в стека
* Ако **командата** е **"remove"**, ще получите **едно число** след командата - **броя на елементите**, които трябва да **премахнете**. Ако няма **достатъчно елементи,** пропуснете командата.

### Изход

* Когато получите "**end**", отпечатайте сумата на останалите **елементи** в **стека**

### Примери

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 2 3 4  adD 5 6  REmove 3  eNd | Sum: 6 |

|  |  |
| --- | --- |
| 3 5 8 4 1 9  add 19 32  remove 10  add 89 22  remove 4  remove 3  end | Sum: 16 |

### Насоки

* Използвайте Stack<int>
* Използвайте методите **Push()**, **Pop()**

## Прост калкулатор

Създайте **прост калкулатор**, който пресмята **прости изрази** с добаване и изваждане. Няма да има скоби.

Използвайте **стек.**

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 2 + 5 + 10 - 2 - 1 | 14 |
| 2 - 2 + 5 | 5 |

### Насоки

* Използвайте **Stack<string>**
* Можете да:
  + добавяте елементи и чрез Pop() да ги извадите
  + или използвате Push() и обърнете стека

## Математически скоби

Ще ви бъде даден аретмичен израз със скоби. Сканирайте низа и извлечете всички подизрази.

Отпечатайте изразите на конзолата.

### Пример

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 1 + (2 - (2 + 3) \* 4 / (3 + 1)) \* 5 | (2 + 3)  (3 + 1)  (2 - (2 + 3) \* 4 / (3 + 1)) |
| (2 + 3) - (2 + 3) | (2 + 3)  (2 + 3) |

### Насоки

* Сканирайте изразите за скоби
  + Ако намерите отваряща скоба, добавете индекса в стеак
  + Ако отркрийте затварящата, махнете най-големия елемент от стека. Този индекс е отварящата скоба.
  + Използвайте сегашната и извадения индекс за да извлечете подизраза

## Горещ картоф

Горещ картоф е игра, която се играе в кръг и започва да се подава горещ картоф. Броенето започва от първото дете. Когато едно дете подаде картоф, то напуска играта.

Създайте програма, която симулира играта горещ картоф. **Отпечатайте всички деца**, които са премахнати от кръга. Накрая отпечатайте последното дете.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Alva James William  2 | Removed James  Removed Alva  Last is William |
| Lucas Jacob Noah Logan Ethan  10 | Removed Ethan  Removed Jacob  Removed Noah  Removed Lucas  Last is Logan |
| Carter Dylan Jack Luke Gabriel  1 | Removed Carter  Removed Dylan  Removed Jack  Removed Luke  Last is Gabriel |

## Задръстване

Напишете програма, която симулира **опашка** на **задръстване**. По време на задръстването има само **N** коли, които могат да **преминат на зелено**. Програмата прочита **колите** като ги добавя в **опашка**. Когато светлината стане зелена трябва да отпечатате колите, които са преминали със следното **съобщение** "{кола} passed!". Програмата спира да работи, когато се въведе командата **"end"**. Накрая трябва да отпечатате **броя на колите**, които са **преминали**.

### Вход

* На **първия ред** ще получите числото **N** - броя на колите, които минават на зелено
* На следващите редове ще получавате команди - низ с кола или с "**green**".
* Програмата спира, когато се въведе "**end**".

### Изход

* Когато се въведе командата **"end"**, отпечатайте "{броя на колите} cars passed the crossroads."

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 4  Hummer H2  Audi  Lada  Tesla  Renault  Trabant  Mercedes  MAN Truck  green  green  Tesla  Renault  Trabant  end | Hummer H2 passed!  Audi passed!  Lada passed!  Tesla passed!  Renault passed!  Trabant passed!  Mercedes passed!  MAN Truck passed!  8 cars passed the crossroads. |
| 3  Enzo's car  Jade's car  Mercedes CLS  Audi  green  BMW X5  green  end | Enzo's car passed!  Jade's car passed!  Mercedes CLS passed!  Audi passed!  BMW X5 passed!  5 cars passed the crossroads. |

## Основни операции с опашка

В тази задача ще се занимавате с опашка. Дадени са ви е числото **N.** То е равно на броя на елементите, които трябва да **добавите (enqueue)**. Числото **S** е равно на **броя числа**, които трябва да се **премахнат** от опашката (**dequeue**). Числото **X**, което трябва да търсите в **опашката** и ако числото е налично, отпечатайте на конзолата **“true”**. В противен случай отпечатайте **най-малкото число**. Ако **няма елементи** в редицата, отпечатайте **“0”**.

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Коментари** |
| 5 2 32  1 13 45 32 4 | true | Имаме да **добавим** 5 елемента. След това **премахваме** 2 елемента. Накрая трябва да проверим дали има 32 в нашата опашка. Тъй като съществува отпечатваме **“true”.** |
| 4 1 666  666 69 13 420 | 13 |  |
| 3 3 90  90 0 90 | 0 |  |

## Максимален и минимален елемент

Имате празна редица от стек и са ви дадени **N** заявки. Всяка заявка може да бъде четири типа:

1 X – **Добавете** елемента X в стека.

2 – **Изтрийте последният** елемент на **стека**.

3 – **Отпечатайте максималното** елемент от стека.

4 – **Отпечатайте минималният** елемент от стека.

След всички заявки принтирайте стека в следният формат:

"{n}, {n1}, {n2} …, {nn}"

### Вход

* На първия ред се чете числото **N**
* На следващите **N** редове ще получавате заявки. (*Гарантирано е, че всяка заявка е валидна.*)

### Изход

* За заявки 3 и 4 принтирайте **максималното/минималното** число от стека на нов ред

### Бележки

* 1 ≤ N ≤ 105
* 1 ≤ x ≤ 109
* Ако **няма** повече елементи в стека, **не отпечатвайте** нищо на команда 3 и 4

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 9  1 97  2  1 20  2  1 26  1 20  3  1 91  4 | 26  20  91, 20, 26 |
| 10  2  1 47  1 66  1 32  4  3  1 25  1 16  1 8  4 | 32  66  8  8, 16, 25, 32, 66, 47 |

## Моден бутик

Притежавате моден бутик и получавате веднъж месечно кутия, която е пълна с дрехи. Трябва да ги подредите в магазина си, така че да вземете от кутията и започнете **от последното облекло** в горната част на купчината до **първото** в дъното. Използвайте **стек**. Всяка дреха има **тегло** (цяло число). Трябва да **сумирате** техните тегла, още докато ги изваждате от кутията. Ще ви бъде дадено число, което е равно на **капацитета** на рафта. **Продължавайте** **да сумирате**, докато сумата е по-малка от капацитета. Ако сумата е равна или по-голяма от капацитета, трябва да вземете нов рафт за следващите дреди, ако има още. Накрая трябва да отпечатате **колко** рафта трябва да се използват за дрехите.

### Вход

* На първия ред ще ви бъде **дадена редица от числа** - дрехите от кутията разделени **с интервал**
* На втория ред ще ви бъде дадено **число** - капацитета на рафтовете

### Изход

* Принтирайте **броя на рафтовете**, които трябва да държат дрехите

### Бележки

* Теглото на всяка дреха ще бъде в интервала [0,20]
* Никога няма да имате повече от 50 дрехи в кутията
* Капацитетът ще бъде число в интервала [0,20]
* **Никое** от тези числа няма да бъде **по-голямо** от **стойността** на **капацитета**

### Пример

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 5 4 8 6 3 8 7 7 9  16 | 5 |
| 1 7 8 2 5 4 7 8 9 6 3 2 5 4 6  20 | 5 |

## Опашка от песни

Напишете програма, която проследява опашка от песни. **Първата** песен, която се добави трябва да бъде **пусната първа**. Не може да се добави песен, ако тя е в опашката.

Ще ви бъде дадена **опашка от песни**, разделени с запетая и интервал. Ще получавате **команди**, докато **има песни в опашката**. Когато вече **няма песни** в опашката, **отпечатайте** "**No more songs!**" и спрете програмата.

Възможните програми са:

* **"Play" - пуска песен (премахва песен от опашката)**
* **"Add {песен}" - добавяте песен към опашката, ако вече е вътре отпечатайте "{песен} is already contained!"**
* **"Show" - принтирайте всички песни от опашката разделени с запетая и интервал (започнете от първата песен в опашката до последната)**

### Вход

* На първия ред ще получите редица от низове разделена с запетая и интервал
* На следващия ред ще ви бъдат давани комданди, докато не свършат песните в опашката

### Изход

* Докато получавате командите, отпечатайте правилните съобщения, описани по-горе
* След командата "Show", прентирайте песните от от първата до последната

### Бележки

* Входа **винаги** ще бъде валиден и във **формата** описан по-горе
* **Може** да има команди **дори след** като **няма песни в опашката** (игнорирайте ги)
* Никога няма да има дублирани песни в първоначалната опашка

### Пример

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| All Over Again, Watch Me  Play  Add Watch Me  Add Love Me Harder  Add Promises  Show  Play  Play  Play  Play | Watch Me is already contained!  Watch Me, Love Me Harder, Promises  No more songs! |

## Камион

Да предположим, че има кръг, в който има **N** на брой бензиностанции. Те са номерирани от 0 до N-1(и двете числа включително). Имате информация съставена от две части за всяка една бензиностанция: (1) **Количеството бензин**, което наливате и (2) **дистанцията** до следващата бензиностанция.

Имате резервоар на камионът с неограничен обем. Можете да стартирате от **всяка** една бензиностанция. Пресметнете от къде да тръгнете, така че да изминете целия кръг. Имайте в предвид, че камионът ще спре на **всяка от бензиностанциите**. Камионът ще се гори по един литър бензин на всеки километър.

### Вход

* На първия ред получавате стойност на **N**
* На следващите **N** реда ще получавате двойки от числа, т.е. количеството бензин, което бензиностанцията ще ви даде, и разстоянието до следващата бензиностанция

### Изход

* Цяло число, което ще бъде най-малкият индекс на бензиностанция, от която можем да започнем обиколката

### Бележки

* 1 ≤ N ≤ 1000001
* 1 ≤ Количество бензин, Дистанция≤ 1000000000

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 3  1 5  10 3  3 4 | 1 |

## Балансиран израз

Дадена ви е последователност от скоби. Вашата задача е да определите дали изразът е **балансиран**. Изразът е балансиран, когато всяка **отворена скоба** е сдвоена **уникално със затворена скоба**. Също така **интервалът** между тях **трябва да бъде балансиран**. Ще ви бъдат дадени **три** типа скоби:(,{ и[.

**{[()]} - Балансиран израз.**

**{[(])} - Не балансиран израз.**

### Вход

* Всеки вход се състои от един ред с **последователност от скоби**.

### Изход

* За всеки ред трябва да отпечатате на нов ред "YES" ако изразът е балансиран. В противен случай "NO"

### Бележки

* 1 ≤ дължината на редицата≤ 1000.
* Всеки символ от редицата **може да бъде** {, }, (, ), [, ].

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| {[()]} | YES |
| {[(])} | NO |
| {{[[(())]]}} | YES |

## Прост редактор

Даден ви е празен низ. Вашата задача е да имплементирате 4 команди, които маниполират низа:

* 1 текст- **Добавя** текст в края на низа.
* 2 N - **Изтрива** последните N на брой елементи
* 3 Индекс - **Връщате** елемента на определен индекс от текста
* 4 - **Отменя** последната команда от тип 1 / 2 и връща текста в състоянието преди тази операция

### Вход

* Първо ще прочитете ***n,*** броя операции.
* Всеки следващ ***n*** ред ще се състой от името на операцията последвана от стойност, ако има такава, разделени с интервал в следния формат “{**Име на командата}** {**Стойност}”.**

### Изход

* На всяка операция от тип ***3*** принтирайте на нов ред върнатия символ.

### Бележки

* 1 ≤ N ≤ 105
* Дължината на текста няма да надвиши 1000000.
* Всички символи ще бъдат на английски.
* Гарантирано е, че последователността от операции за въвеждане е възможна за изпълнение.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 8  1 abc  3 3  2 3  1 xy  3 2  4  4  3 1 | c  y  a |

### Обяснение

* Имаме 8 операции. Низа първоначално е празен.
* На следващата операция добавяме текста **“abc”**.
* След това принтираме третата буква, която е **c**.
* Изтриваме трите букви - **“abc”**.
* Добавяме **“xy”** към низа.
* Изтриваме **“xy”** от низа
* Връщаме **“xy”**
* След това принтираме втората буква, която е **y**.
* Връщаме **“abc”**
* След това принтираме третата буква, която е **c**.

## \*Ключов револвер

Нашият любим супер шпионин Сам се е върнал от мисия. Този път има по-трудна задача. Трябва да **отключи сейф**. Проблема е, че сейфът е **заключен** от **няколко ключалки**, като всички са с **различни размери**.

Нашият герой има специално оръжие, което се нарича **ключов револвер** със специални патрони. Всеки **патрон** може да отключи **ключалка** с размер **равен или по-голям** от **патрона**. Патронът влиза в ключалката и взривява **всичко**. Сам **не знае размера** на ключалките, затова трябва да стерля по тях докато не свършат ключалките.

Може би се питате какво има в сейфа? Говори се, че заклетият враг на Сам - **Николадзе**, държи вътре своята рецепта за **грузинска ракия Чача**. Едно последно нещо, всеки патрон, който Сам изтрелва ще му струва пари, **които ще бъдат приспаднати от заплатата му** за разузнаването.

### Вход

* На **първия ред** ще получите цените на всички **патрони** - **числа в обхват** **[0-100]**
* На **втория ред** ще получите **размерите на патроните** - **числа в обхват [1-5000]**
* На **третия ред** ще получите **патроните** - **числа разделени с интервал в обхват [1-100]**
* На **четвъртия ред** ще получите **ключалките** - **числа разделени с интервал в обхват [1-100]**
* На **петия ред** ще получите **заплатата му** - **число в обхват [1-100000]**

След като Сам е получил информацията и екипировката (**входа**), той **започва да стреля по ключалките** отпред назад, докато патроните минават през ключалките **отзад напред**.

Ако **патронът** е **по-малък или равен** на **сегашната ключалка** отпечатайте **“Bang!”**, след това **премахнете ключалката**. В противен случай отпечатайте **"**Ping!**"**, оставяйки ключалката **непокътната**. И в двата случия **коршумите се премахват**.

Ако Сам остане без патрони на определен ключалка отпечатайте **"**Reloading!**"** на конзолата, след продължавате да стреляте. Ако няма повече патрони, **не го** отпечатвайте.

Програмата спира, когато Сам **няма повече патрони** или сейфът **няма повече ключалки**.

### Изход

* Ако Сам няма повече патрони преди ключалките на сейфът да свършат, отпечатайте:

**"**Couldn't get through. Locks left: {броя на оставащите ключалки}**"**

* Ако Сам **успее да** отвори сейфът, отпечатайте:  
  **"**{броя на оставащите ключалки коршуми} bullets left. Earned ${спечелините пари}**"**

Не забравяйте да вземете в **предвид цената на куршумите**, когато изчислявате **спечелените пари**.

### Бележки

* Входът **винаги ще бъде валиден**. **Няма нужда** да го проверявате изрично.
* Никога **няма** да има случай, в който Сам разбива ключалката и завършва с **отрицателен баланс**.

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Коментари** |
| 50  2  11 10 5 11 10 20  15 13 16  1500 | Ping!  Bang!  Reloading!  Bang!  Bang!  Reloading!  2 bullets left. Earned $1300 | 20 стреля ключалка 15 (ping)  10 стреля ключалка 15 (bang)  11 стреля ключалка 13 (bang)  5 стреля ключалка 16 (bang)  Цената на коршумите:  4 \* 50 = $200  Спечелени:  1500 – 200 = $1300 |
| 20  6  14 13 12 11 10 5  13 3 11 10  800 | Bang!  Ping!  Ping!  Ping!  Ping!  Ping!  Couldn't get through. Locks left: 3 | 5 стреля ключалка 13 (bang)  10 стреля ключалка 3 (ping)  11 стреля ключалка 3 (ping)  12 стреля ключалка 3 (ping)  13 стреля ключалка 3 (ping)  14 стреля ключалка 3 (ping) |

|  |  |  |
| --- | --- | --- |
| 33  1  12 11 10  10 20 30  100 | Bang!  Reloading!  Bang!  Reloading!  Bang!  0 bullets left. Earned $1 | 10 стреля ключалка 10 (bang)  11 стреля ключалка 20 (bang)  12 стреля ключалка 30 (bang)  Цената на коршумите:  3 \* 33 = $99  Спечелени:  100 – 99 = $1 |

## \*Чаши и шишета

Ще ви бъде дадена **редица от числа** - всяко число съдържа **капацитета на чаша**. На нови ред ще получите **още една редица от числа** - обема на **шишета с вода**. Вашата задача е да **напълните** чашите.

Пълненето става чрез избиране на **точно една** бутилка. Трябва да вземете **последната получена ботилка** и започвате да пълните от **първата въведена чаша**. Ако ботилката има **N** вода, вие **намалята** свободното количество на първата чаша с **N**.

Когато **количеството** на чашата достигне **0 или по-малко**, се **премахва**. **Възможно** е количеството на чашата да е **по-голямо** от количеството в ботилката. **В такъв случай** взимаме друга ботилка, докато чашате достигне 0 или повече. Ако ботилката е **по-голяма или равна** на чашата, трябва да напълвате чашата и останалата вода **се губи**. Трябва да **следите изразходваните литри вода** и ги отпечатайте в края на програмата.

Ако **сте успели да напълните всички чаши**, отпечатайте **останалите бутилки с вода**, от **последната въведена до първата**, в противен случай трябва да отпечатате **останалите чаши** – **от първата до последната**.

### Вход

* На **първия ред** ще получите редица от числа- **капацитета** на чашите разделени с **интервал.**
* На **втория ред** ще получите редица от числа - ботилките запълнени с вода разделени с **интервал**.

### Изход

* На първия ред трябва да отпечатате ботилките или оставащите чаши във взависимост от случия. **Спазвайте реда** за **печат точно както е посочено**.
  + **"Bottles: {remainingBottles}"** or **"Cups: {remainingCups}"**
* На втория ред принтирайте изхабените литри от вода в следния формат:**"Wasted litters of water: {общото количество вода изхъбено}".**

### Бележки

* Всички числа ще бъдат в обхвата [1, 500].
* **НЯМА** да има случай, в който водата да е точно толкова, колкото са стойностите на чашите, така че накрая да няма чаши и вода в бутилките
* Маскимално време : 100ms
* Маскимална памет : 16MB

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Comment** |
| 4 2 10 5  3 15 15 11 6 | Bottles: 3  Wasted litters of water: 26 | Вземеме първата чаша и последната ботилка, както е описано в условието.  6 – 4 = 2 – имаме още 2, така че похабената вода става 2.  11 – 2 = 9 – отново е повече, така че го добавяме към предишната сума, която е 2 и става 11.  15 – 10 = 5 – изразходваната вода става 16.  15 – 5 = 10 – изразходваната вода става 26.  Успяхме да напълним всички чаши, така че отпечатваме останалите бутилки и общото количество изхабена вода. |
| 1 5 28 1 4  3 18 1 9 30 4 5 | Cups: 4  Wasted litters of water: 35 |  |
| 10 20 30 40 50  20 11 | Cups: 30 40 50  Wasted litters of water: 1 |  |