# Упражнения: Стек и опашка

Тествайте решението в Judge: <https://judge.softuni.org/Contests/4153/03-Stack-and-Queue-Basics>.

## Обърнат низ

Напишете програма, която:

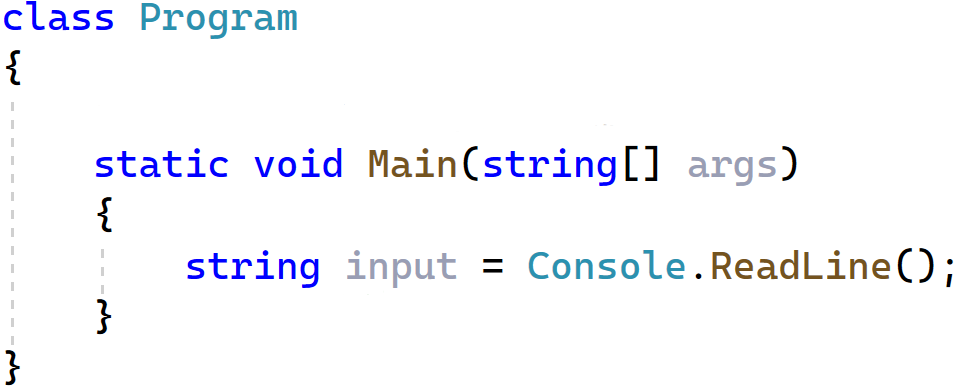
* **Прочита** низ
* **Обръща го** като **използва** Stack<T>
* **Отпечатва** резултата

### Примери

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

### Насоки

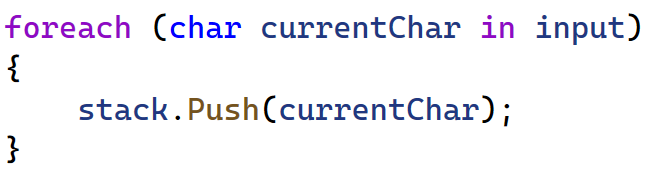
1. Първо трябва да прочетете **входа**:



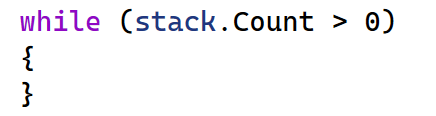
1. След това трябва да създадете празен **стек** от **символи** с име **stack**:



1. Обходете всеки един **символ** от входа и го **добавете** в **стека**:



1. Създайте **while-цикъл**. Той трябва да итерира докато броя на символите в стека е **по-голям** от **0**:



1. В неговото тяло отпечатайте елементите с командата **Pop()**.



## Сума от стек

Напишете програма, която:

* **Чете** вход от числа и ги **добавя в стек**
* **Чете команди** докато не получите **"end".** Командите могат да бъдат:
  + add **-**  ще получите **две числа** след командата, които трябва да **добавите** в стека
  + remove - ще получите **едно число** след командата - **броя на елементите**, които трябва да **премахнете**. Ако няма **достатъчно елементи,** пропуснете командата.
* **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"**
* Всички команди могат да бъдат с **малки** или **големи** **букви**

### Изход

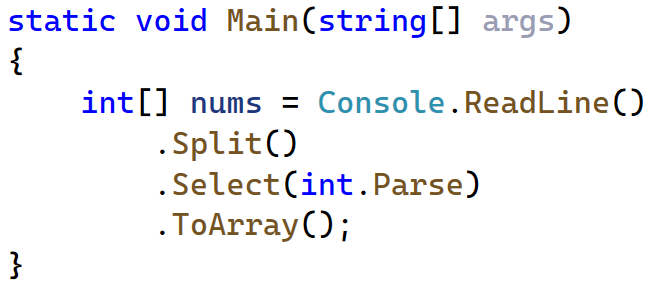
* Когато получите "**end**", отпечатайте сумата на останалите **елементи** в **стека**

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 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

1. Прочетете **входа**:



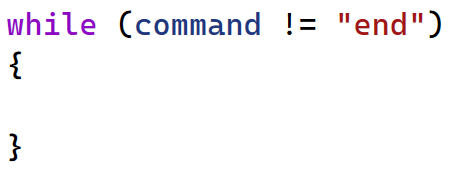
1. Създайте стек с име **numStack**, в който да сложите числа:



1. Инициализирайте стринг **command**, чрез който да четете командите:



1. Създайте **while-цикъл**, който итерира до достигане на команда **"end"**:

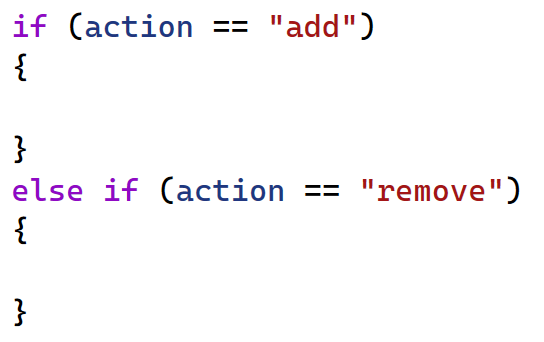


1. В цикъла разбийте стринга с командата на **масив** от елементи (**токени**). Първият токен показва **действието** ("**add**" или "**remove**"):

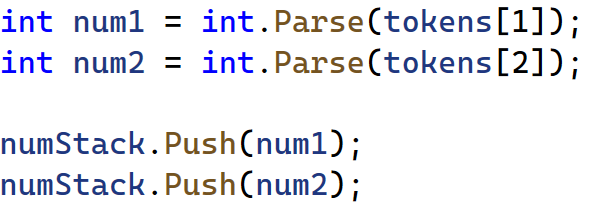
A black and blue text

Description automatically generated

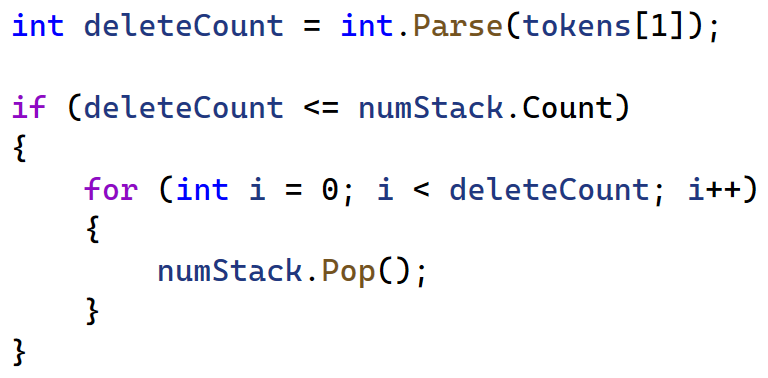
1. В цикъла добавете **if-else** или **switch-case** конструкция за различните команди. Създайте кейсове за **"add"** и "remove":



1. Имплементирайте логиката за **"**add**"**:

****

1. Имплементирайте логиката за **"**remove**"**:



1. След **условната конструкция** прочетете нова команда (**command**):



1. След като **приключите** с четенето на команди, отпечатайте **сумата на стойностите** в стека:



## Прост калкулатор

Създайте **прост калкулатор**, който пресмята **прости изрази** със събиране 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 чрез **Push()** и чрез Pop() да ги изваждате.
  + или използвате Push() и обърнете стека.

## Математически скоби

Ще ви бъде даден **аретмичен израз със скоби**. Анализирайте израза и **извлечете всички подизрази**.

**Отпечатайте** тези подизрази 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 **stack.**
  + Ако отркрийте **затваряща скоба**, махнете **последно добавения** елемент от стека. Това е **индексът** на съответстващата **отваряща скоба**
  + На база на **индексите** на **отварящата** и на **затварящата** скоба извлечете съответния **подизраз**.

## 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. |

## 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. От конзолата ще получите низ с **имената на децата**, а на следващия ред – число **n**. При всяко **n­-**то хвърляне, дете се **отстранява от играта**, докато остане само едно. **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 |