# Упражнения: Стек и опашка

Тествайте решението в Judge: <https://judge.softuni.org/Contests/4154/03-Stack-and-Queue-Advanced>.

## Основни операции с опашка

В тази задача ще се занимавате с опашка. Дадено 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**). **X** е число, 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** | **Обяснение** |
| 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 |  |

### 00Насоки

1. Прочетете числата N, S и X от конзолата.
2. Създайте **опашка** за съхранение на **елементите**.
3. Извършете операциите за **добавяне** (Enqueue()) и **премахване** (Dequeue()) на елементи от опашката.
4. Използвайте метода Contains(), за да проверите дали X е в опашката. Ако стойността на X е налична, отпечатайте **"**true**"**. В противен случай, отпечатайте **най-малкото число** в опашката чрез метода Min().

## Максимален и минимален елемент

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

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

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

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

4 – **Отпечатайте минималния** 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 |

### Насоки

1. Прочетете **броя на заявките**.
2. Създайте **стек**, върху които ще **извършвате операциите**.
3. Създайте for**-цикъл**, за да обходите всички заявки.
4. За заяква **"1"** използвайте метода Push(), за да **добавите число** в **стека**.
5. За заявка **"2"** използвайте метода Pop(), за да **премахнете** **последния** **елемент** от **стека**.
6. За заявка **"3"** използвайте метода Max(), за да **принтирате** **най-голямото число**.
7. За заявка **"4"** използвайте метода Min(), за да **принтирате** **най-малкото число**.
8. След като приключите със заявките, **отпечатайте** **числата** в **стека**.

## Моден бутик

Притежавате моден бутик и получавате веднъж месечно кутия, която е пълна с дрехи. Трябва да ги подредите в магазина си, така че първо да вземете от кутията **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 дрехи, 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**

### Примери

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

### Насоки

1. Прочете **стека**.
2. Прочетете числото, което представлява **максималния капацитет на рафта**.
3. Създайте **брояч** за **използваните рафтове**.
4. Създайте още една променлива, която ще използвате за **капацитета на текущия рафт** със стойност **максималния капацитет на рафта**.
5. Създайте while**-цикъл**, за да обходите всички дрехи.
6. В него вземете **последната дреха** от **стека**.
7. Проверете дали **дрехата** може да бъде **добавена към рафта**.
8. Ако **може да бъде добавена**, **премахнете дрехата** от **стека** и намалете **теглото на текущия рафт** с **теглото на дрехата**.
9. В противен случай, задайте стойност на капацитета на текущия рафт **максималния капацитет** и **увеличете използваните** **рафтове** с **едно**.
10. Накрая отпечатайте броя на **използваните рафтове**.

## 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! |

## Супермаркет

Ще получавате **имена**. Трябва да ги добавите в **опашка**, докато не получите "**End**". Ако получите **"Paid"**, отпечатайте имената и изчистете опашката. В противен случай ще получавате имена. Когато получите **"End",** трябва да отпечатате броя на оставащите хора в следния формат: "**{брой} people remaining.".**

### Примери

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Вход** | **Изход** |
| Liam  Noah  James  Paid  Oliver  Lucas  Logan  Tiana  End | Liam  Noah  James  4 people remaining. | Amelia  Thomas  Elias  End | 3 people remaining. |

## Бързо хранене

Имате магазин за бързо хранене. Повечето храна, която имате е предварително подготвена. Трябва да знаете дали вашата храна ще стигне за всички клиенти. Напишете програма, която проверява количеството на поръчките. Искате да намерите клиента с **най-голямата поръчка**, защото ще му дадете отстъпка за следващият път.

Първо ще получите количество храна, която имате за деня (число). След това ще получите редица от числа, количеството на всяка поръчка. Запазвате поръчките в опашка. Намирате най-голямата поръчка и я отпечатайте. Ще започнете да обслужвате от първия клиент. Преди това проверете дали имате достатъчна храна. Ако имате, премахнете поръчката от опашката и намалете количеството храна, което имате. Ако успешно сте обслужели клиента, отпечатайте : "Orders complete". В противен случай отпечатайте: "Orders left: {поръчка1} {поръчка2} .... {поръчкаN}".

### Вход

* На първия ред ще получите количеството храна, с което разполагате - **число** в обхвата [0, 1000]
* На втория ред ще получите редица от числа, **разделени с интервал** - поръчките

### Изход

* Отпечатайте количеството на най-голямата поръчка
* Отпечатайте "Orders complete", ако поръчките са готови
* Ако има останали поръчки, отпечатайте тях във формата по-горе

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 348  20 54 30 16 7 9 | 54  Orders complete |
| 499  57 45 62 70 33 90 88 76 | 90  Orders left: 76 |