Упражнения: Стекове

1. Обръщане на числа със стек

Напишете програма, която чете **N цели числа** от конзолата и **ги обръща в обратен на въвеждането ред, чрез стек**. Използвайте **Stack<int>** класа от .NET Framework. Просто поставете (**put)** въвежданите числа в стека и ги вземете (**pop)** после от стека.

Примери

Вход	Изход
1 2 3 4 5	5 4 3 2 1
1	1
(empty)	(empty)
1 -2	-2 1

2. Всички цифри и сумата им

Напишете програма, която чете **цяло число** от конзолата и извежда **всичките му цифри** и **сумата** им.

Примери

Вход	Изход
5368	5 + 3 + 6 + 8 = 22
10	1 + 0 = 1
108	1 + 0 + 8 = 9

3. Казвано ли е дадено число?

Напишете програма, която чете от конзолата **брой N** и после **последователност от N цели числа**, всяко на отделен ред и накрая число, което се проверява дали съществува в първата група числа**.** Ако числото е сред тях, се извежда **"{number} exists in the List"**, в противен случай - **"{Number} not exists in the List"**. Програмата трябва да ползва стек за съхранение на числата, но без да вика вградения метод Contains.

4. Отделно положителните, отделно отрицателните

Напишете програма, която чете от конзолата **редица от цели числа.** Числата са подадени на един ред, разделени с интервал. Трябва да се изведат два реда - на първия всички положителни числа от горната редица, а на втория всички отрицателни числа - в реда, в който са подадени, отново разделени с интервал. За съхранение на двете групи числа да се ползват стекове.

5. Сортиране чрез стек

Напишете програма, която чете от конзолата **редица от цели числа.** Числата са подадени на един ред, разделени с интервал. Трябва да се изведе на един ред **същата редица, подредена във възходящ ред**, числата да отново отделени с интервал. Задачата да бъде решена, като за подредената редица бъде използвана само структура от данни стек. Масиви и списъци не е разрешено да се ползват за сортирането, а само за съхраняване на входните данни.

6. Сума на много големи числа

Напишете програма, която чете от конзолата **две много големи положителни цели числа**, всяко указано на отделен ред. Числата са указани без знак и може да са с повече цифри, отколкото може да побере decimal типа, така че ги прочетете в низ. Трябва да се изведе **сумата** им. За съхраняване на цифрите използвайте стекове.

7. Обработка на текст

Напишете програма, която приема като входни данни на първи ред символен низ. На следващия ред получава следните команди в такъв формат:

Append <Символен низ>

Remove <pos> <number>

Insert <pos> <string>

Replace <substring> <substring>

Undo

End

Накрая извежда резултатния символен низ.

Вход

Входът ще се състои от **множество редове**:

- На първия ред, ще бъде символния низ, който ще се променя
- На втори ред ще бъде въведена команди с параметрите в указания формат
- Обработката им трябва да продължава до срещане на команда End

Изход

Отпечатате изречението, след обработката.

Примери

Вход	Изход
That is not true. Remove 8 4 End	`That is true.`
That is not true. Remove 8 4 Undo	`That is not OK.`

Replace true OK End	
I'm still alive! Remove 4 6 Replace alive dead Remove 0 9 Insert 0 Finito! Undo Undo Undo Undo End	I'm still alive!

8. Обединяване на подредени редици

Напишете програма, която чете от конзолата **брой N** и после **последователност от N двойки цели числа**, всяка на отделен ред. Гарантирано е, че числата от първата колонка са във възходящ ред. За втората важи същото, но първото число от всяка двойка може да е по-голямо от второто, равно или по-малко от него. Да се обединят всички числа така, че **новополучената редица отново да е възходящо подредена**. **Изведете новополучената редица** на един ред, числата да са отделени с интервали. За съхранение на числата използвайте сам стекове.

9. Статична имплементация на стек

Имплементирайте статично стек **`Stack<T>`**, който пази елементите си в масив:

```
public class ArrayStack<T>
{
    private T[] elements;
    public int Count { get; private set; }
    private const int InitialCapacity = 16;
    public ArrayStack(int capacity = InitialCapacity) { … }
    public void Push(T element) { … }
    public T Pop() { … }
    public T[] ToArray() { … }
    private void Grow() { … }
}
```

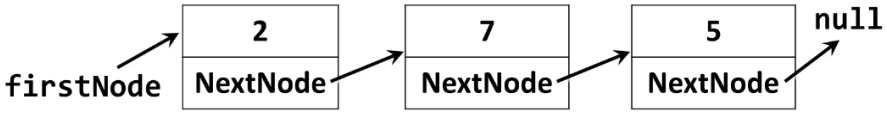
Подсказки:

- Капацитета на стека е **`this.elements.Length`**
- Пазете размера на стека (брой елементи) в **`this.Count`**
- **`Push(element)`** запазва **`елемента`** в **`elements[this.Count]`** и увеличава **`this.Count`**
- **`Push(element)`** трябва да извика **`Grow(),`** в случай че **`this.Count == this.elements.Length`**
- **`Pop()`** намаля **`this.Count`** и връща **`this.elements[this.Count]`**

- **`Grow()`** заделя нов масив **`newElements`** с размер **`2 * this.elements.Length`** и копира първите **`this.Count`** елемента от **`this.elements`** до **`newElements`**. Накрая, присвоете **`this.elements = newElements`**
- **`ToArray()`** създава и връща масив от **`this.elements[0…this.Count-1]`**
- **`Pop()`** трябва да хвърля **`InvalidOperationException`** (или **`IllegalArgumentException`**) при празен стек

10. Имплементиране на свързан стек

Имплементирайте стек чрез "**свързан списък**":



Използвайте следния код като за начало:

```
public class LinkedStack<T>
{
    private Node<T> firstNode;
    public int Count { get; private set; }
    public void Push(T element) { … }
    public T Pop() { … }
    public T[] ToArray() { … }
    private class Node<T>
    {
        private T value;
        public Node<T> NextNode { get; set; }
        public Node(T value, Node<T> nextNode = null) { … }
    }
}
```

`Push(element)` операцията трябва да създаде нов **`Node<T>`** и да го зададе като **`firstNode`**: **`this.firstNode = new Node<T>(element, this.firstNode)`**.

`Pop()` операцията трябва да върне **`firstNode`** и да го замени с **`firstNode.NextNode`**. Ако стекът е празен, то трябва да се хвърли **`InvalidOperationException`**.