Lab: Arrays

Day of Week

Enter a day number [1…7] and print the name (in English) or "Invalid day!"

Examples

Input	Output
1	Monday
2	Wednesday
10	Invalid day!

Print Numbers in Reverse Order

Read n numbers and print them in reverse order.

Examples

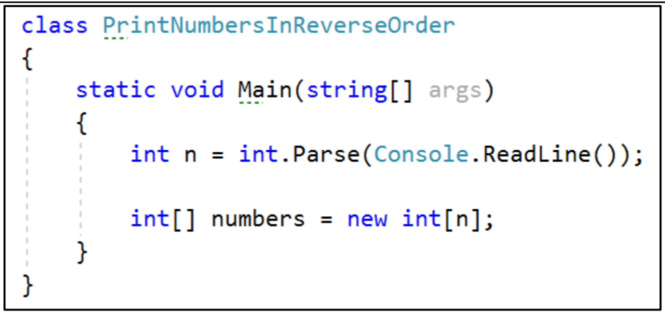
Input	Output
3 10 20 30	30 20 10
3 30 20 10	10 20 30
1 10	10

Hints

First, we need to read **n** from the console.

```
class PrintNumbersInReverseOrder
{
    static void Main(string[] args)
    {
        int n = int.Parse(Console.ReadLine());
    }
}
```

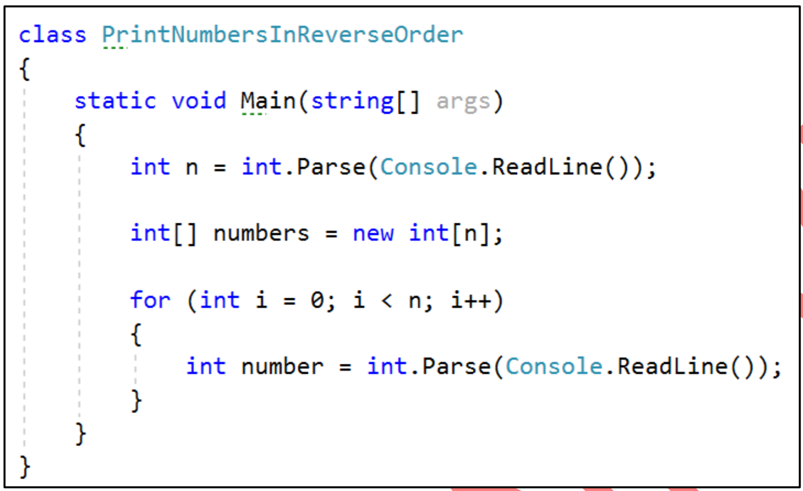
Create an **array of integer** with **n** size.



```
class PrintNumbersInReverseOrder
{
    static void Main(string[] args)
    {
        int n = int.Parse(Console.ReadLine());

        int[] numbers = new int[n];
    }
}
```

Read **n** numbers using for loop.

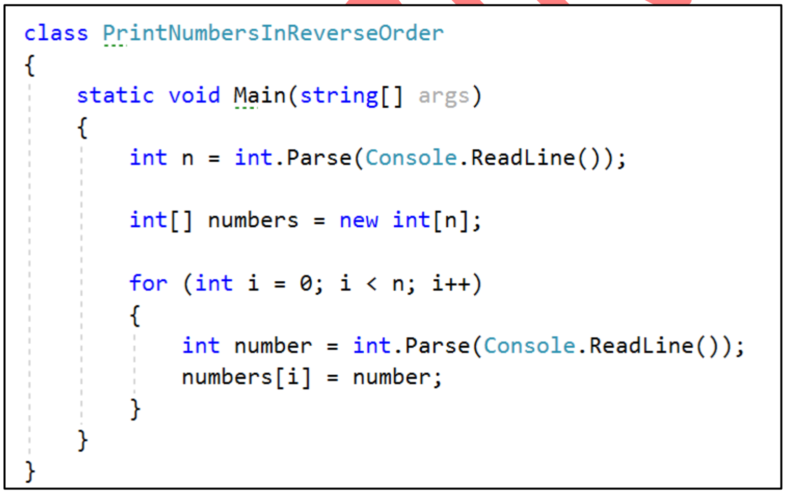


```
class PrintNumbersInReverseOrder
{
    static void Main(string[] args)
    {
        int n = int.Parse(Console.ReadLine());

        int[] numbers = new int[n];

        for (int i = 0; i < n; i++)
        {
            int number = int.Parse(Console.ReadLine());
        }
    }
}
```

Set number to the corresponding **index**.

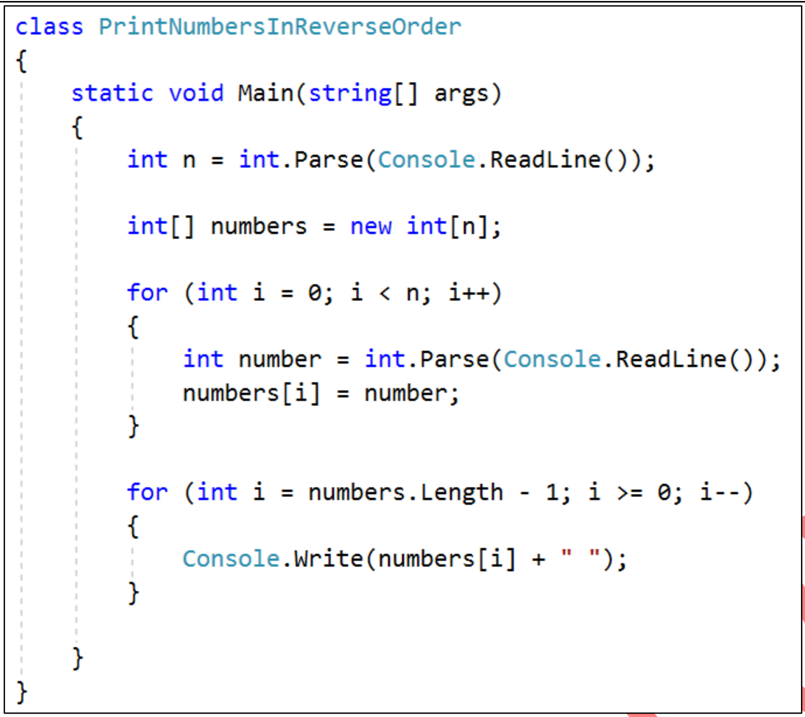


```
class PrintNumbersInReverseOrder
{
    static void Main(string[] args)
    {
        int n = int.Parse(Console.ReadLine());

        int[] numbers = new int[n];

        for (int i = 0; i < n; i++)
        {
            int number = int.Parse(Console.ReadLine());
            numbers[i] = number;
        }
    }
}
```

Print the array in reversed order.



```
class PrintNumbersInReverseOrder
{
    static void Main(string[] args)
    {
        int n = int.Parse(Console.ReadLine());

        int[] numbers = new int[n];

        for (int i = 0; i < n; i++)
        {
            int number = int.Parse(Console.ReadLine());
            numbers[i] = number;
        }

        for (int i = numbers.Length - 1; i >= 0; i--)
        {
            Console.Write(numbers[i] + " ");
        }

    }
}
```

Rounding Numbers

Read an array of real numbers (space separated), round them in "**away from 0**" style and print the output as in the examples:

Examples

Input	Output

`0.9 1.5 2.4 2.5 3.14`	`0.9 => 1`
	`1.5 => 2`
	`2.4 => 2`
	`2.5 => 3`
	`3.14 => 3`

`-5.01 -1.599 -2.5 -1.50 0`	`-5.01 => -5`
	`-1.599 => -2`
	`-2.5 => -3`
	`-1.50 => -2`
	`0 => 0`

Reverse Array of Strings

Read an array of strings (space separated values), reverse it and print its elements:

Examples

Input	Output
`a b c d e`	`e d c b a`

-1 hi ho w	w ho hi -1

Sum Even Numbers

Read an array from the console and sum only the even numbers.

Examples

Input	Output
1 2 3 4 5 6	12
3 5 7 9	0
2 4 6 8 10	30

Hints

First, we need to read the array.

```
class SumEvenNumbers
{
    static void Main(string[] args)
    {
        int[] numbers = Console.ReadLine()
            .Split()
            .Select(int.Parse)
            .ToArray();
    }
}
```

We will need a variable for the sum.

```
int sum = 0;
```

Iterate through all elements in the array with for loop.

```
for (int i = 0; i < numbers.Length; i++)
{
}
```

Check if the number at current index is even.

```
for (int i = 0; i < numbers.Length; i++)
{
    int currentNumber = numbers[i];
    if (currentNumber % 2 == 0)
    {
        sum += currentNumber;
    }
}
```

Print the total sum

```
Console.WriteLine(sum);
```

Even and Odd Subtraction

Write a program that calculates the difference between the sum of the even and the sum of the odd numbers in an array.

Examples

Input	Output	Comments
1 2 3 4 5 6	3	Even: 2 + 4 + 6 = 12 Odd: 1 + 3 + 5 = 9 Result: 12 – 9 = 3
3 5 7 9	-24	Even: 0 Odd: 3 + 5 + 7 + 9 = 24 Result: 0 – 24 = -24
2 4 6 8 10	30	Even: 2 + 4 + 6 + 8 + 10 = 30 Odd: 0 Result: 30 – 0 = 30

Hints

First, we need to read the array.

```
class EvenOddSubtraction
{
    static void Main(string[] args)
    {
        int[] numbers = Console.ReadLine()
            .Split()
            .Select(int.Parse)
            .ToArray();
    }
}
```

We will need two variables – even and odd sum.

```
int evenSum = 0;
int oddSum = 0;
```

Iterate through all elements in the array with for loop.

```
for (int i = 0; i < numbers.Length; i++)
{
}
```

Check the current number – if it is even add it to the even sum, otherwise add It to the odd sum.

```
int currentNumber = numbers[i];
if (currentNumber % 2 == 0)
{
    evenSum += currentNumber;
}
else
{
    //TODO
}
```

Print the difference.

```
int differene = evenSum - oddSum;
Console.WriteLine(differene);
```

Equal Arrays

Read two arrays and print on the console whether they are identical or not. Arrays are identical if their elements are equal. If the arrays are identical find the sum of the first one and print on the console following message: "Arrays are identical. Sum: {sum}", otherwise find the first index where the arrays differ and print on the console following message: "Arrays are not identical. Found difference at {index} index".

Examples

Input	Output
10 20 30 10 20 30	Arrays are identical. Sum: 60
1 2 3 4 5 1 2 4 3 5	Arrays are not identical. Found difference at 2 index
1 10	Arrays are not identical. Found difference at 0 index

Hints

First, we need to read two arrays.

```
class EqualArrays
{
    static void Main(string[] args)
    {
        int[] arr1 = Console.ReadLine()
            .Split()
            .Select(int.Parse)
            .ToArray();

        int[] arr2 = Console.ReadLine()
            .Split()
            .Select(int.Parse)
            .ToArray();
    }
}
```

Iterate through arrays and compare element. If the elements are not equal print the required message and break the loop.

```
for (int i = 0; i < arr1.Length; i++)
{
    if (arr1[i] != arr2[i])
    {
        Console.WriteLine($"Arrays are not identical. Found difference at {i} index");
        break;
    }
}
```

Think about how to solve the other part of the problem.

Condense Array to Number

Write a program to read **an array of integers** and **condense** them by **summing** adjacent couples of elements until a **single integer** is obtained. For example, if we have 3 elements {2, 10, 3}, we sum the first two and the second two elements and obtain {2+10, 10+3} = {12, 13}, then we sum again all adjacent elements and obtain {12+13} = {25}.

Examples

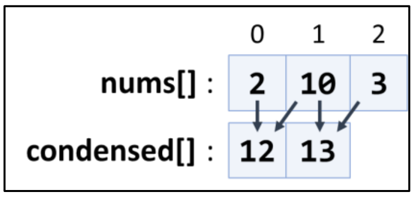
Input	Output	Comments
2 10 3	25	2 10 3 → 2+10 10+3 → 12 13 → 12 + 13 → 25
5 0 4 1 2	35	5 0 4 1 2 → 5+0 0+4 4+1 1+2 → 5 4 5 3 → 5+4 4+5 5+3 → 9 9 8 → 9+9 9+8 → 18 17 → 18+17 → 35
1	1	1 is already condensed to number

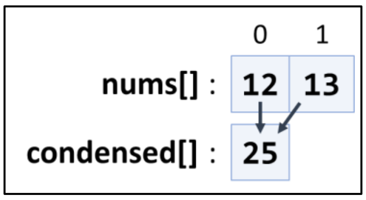
Hints

While we have more than one element in the array **`nums[]`**, repeat the following:

- Allocate a new array **`condensed[]`** of size **`nums.Length-1`**.
- Sum the numbers from **`nums[]`** to **`condensed[]`**:
    - **`condensed[i] = nums[i] + nums[i+1]`**
- **`nums[] = condensed[]`**

The process is illustrated below:





Train

You will be given a count of wagons in a train **n**. On the next **n** lines you will receive how many people are going to get on that wagon. At the end print the whole train and after that the sum of the people in the train.

Examples

Input	Output
3 13 24 8	13 24 8 45
6 3 52 71 13 65 4	3 52 71 13 65 4 208
1 100	100 100

Common Elements

Write a program, which prints common elements in two arrays. You have to compare the elements of the second array to the elements of the first.

Examples

Input	Output
Hey hello 2 4 10 hey 4 hello	4 hello
S of t un i of i 10 un	of i un
i love to code code i love to	code i love to

Zig-Zag Arrays

Write a program which creates 2 arrays. You will be given an integer **n**. On the next **n** lines you get 2 integers. Form 2 arrays as shown below.

Examples

Input	Output

4
1 5
9 10
31 81
41 20
	1 10 31 20
	5 9 81 41

2
80 23
31 19
	80 19
	23 31

Array Rotation

Write a program that receives an array and number of rotations you have to perform (first element goes at the end) Print the resulting array.

Examples

Input	Output
51 47 32 61 21 2	32 61 21 51 47
32 21 61 1 4	32 21 61 1
2 4 15 31 5	4 15 31 2

Top Integers

Write a program to find all the top integers in an array. A top integer is an integer which is **bigger** than all the elements to its right.

Examples

Input	Output
1 4 3 2	4 3 2
14 24 3 19 15 17 17	24 19 17
27 19 42 2 13 45 48	48

Equal Sums

Write a program that determines if there **exists an element in the array** such that the **sum of the elements on its left** is **equal** to the **sum of the elements on its right**. If there are **no elements to the left / right**, their **sum is considered to be 0**. Print the **index** that satisfies the required condition or **"no"** if there is no such index.

Examples

Input	**Output**	**Comments**
1 2 3 3	2	At a[2] -> left sum = 3, right sum = 3 a[0] + a[1] = a[3]
1 2	no	At a[0] -> left sum = 0, right sum = 2 At a[1] -> left sum = 1, right sum = 0 No such index exists
1	0	At a[0] -> left sum = 0, right sum = 0
1 2 3	no	No such index exists
10 5 5 99 3 4 2 5 1 1 4	3	At a[3] -> left sum = 20, right sum = 20 a[0] + a[1] + a[2] = a[4] + a[5] + a[6] + a[7] + a[8] + a[9] + a[10]

Max Sequence of Equal Elements

Write a program that finds the **longest sequence of equal elements** in an array of integers. If several longest sequences exist, print the leftmost one.

Examples

Input	**Output**
2 1 1 2 3 3 **2 2 2** 1	2 2 2
1 1 1 2 3 1 3 3	1 1 1
4 4 4 4	4 4 4 4

0 **1 1** 5 2 2 6 3 3	1 1

Magic Sum

Write a program, which prints all unique pairs in an array of integers whose sum is equal to a given number.

Examples

Input	Output
1 7 6 2 19 23 8	1 7 6 2

14 20 60 13 7 19 8
27
	14 13
	20 7
	19 8

Lab: Lists

Remove Negatives and Reverse

Read a **list of integers**, **remove all negative numbers** from it and print the remaining elements in **reversed order**. In case of no elements left in the list, print "**empty**".

Examples

Input	Output
`10 -5 7 9 -33 50`	`50 9 7 10`
`7 -2 -10 1`	`1 7`
`-1 -2 -3`	`empty`

Solution

Read a list of integers.

```
List<int> numbers = Console.ReadLine()
    .Split()
    .Select(int.Parse)
    .ToList();
```

Remove all negative numbers.

```
numbers.RemoveAll(n => n < 0);
```

If the list count is equal to 0 print "empty", otherwise print all numbers joined by space.

```
if (numbers.Count == 0)
{
    Console.WriteLine("empty");
}
else
{
    Console.WriteLine(string.Join(" ", numbers));
}
```

Sum Adjacent Equal Numbers

Write a program to **sum all adjacent equal numbers** in a list of decimal numbers, starting from **left to right**.

- After two numbers are summed, the obtained result could be equal to some of its neighbors and should be summed as well (see the examples below).
- Always sum the **leftmost** two equal neighbors (if several couples of equal neighbors are available).

Examples

Input	Output	Explanation
3 3 6 1	12 1	**3 3** 6 1 → **6 6** 1 → 12 1
8 2 2 4 8 16	16 8 16	8 **2 2** 4 8 16 → 8 **4 4** 8 16 → **8 8** 8 16 → 16 8 16
5 4 2 1 1 4	5 8 4	5 4 2 **1 1** 4 → 5 4 **2 2** 4 → 5 **4 4** 4 → 5 8 4

Solution

Read a list from numbers.

```
List<double> numbers = Console.ReadLine()
    .Split()
    .Select(double.Parse)
    .ToList();
```

Iterate through the elements. Check if the number at the current index is equal to the next number. If it is, aggregate the numbers and reset the loop, otherwise don't do anything.

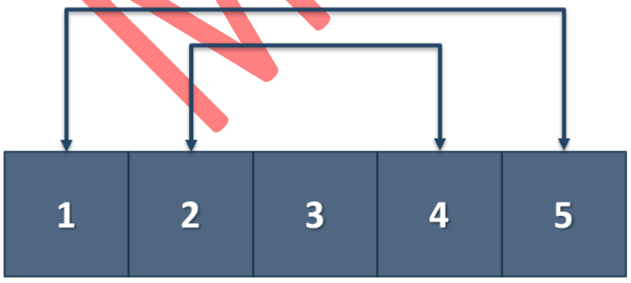
```
if (numbers[i] == numbers[i + 1])
{
    numbers[i] += numbers[i + 1];
    numbers.RemoveAt(i + 1);
    i = -1;
}
```

Finally, you have to print the numbers joined by space.

```
Console.WriteLine(string.Join(" ", numbers));
```

Gauss' Trick

Write a program that **sum** all **numbers in a list** in the following order:
first **+** last, first + 1 **+** last - 1, first + 2 **+** last - 2, … first + n, last - n.



Example

Input	Output
1 2 3 4 5	6 6 3
1 2 3 4	5 5

List of Products

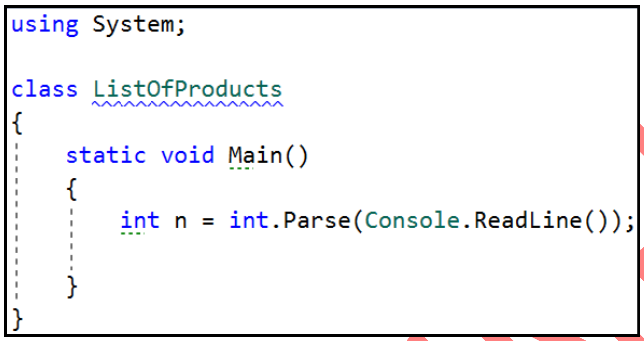
Read a number **n** and **n lines of products**. Print a **numbered list** of all the products **ordered by name**.

Examples

Input	Output
4 Potatoes Tomatoes Onions Apples	 1.Apples 2.Onions 3.Potatoes 4.Tomatoes

Solution

First, we need to read the number **n** from the console

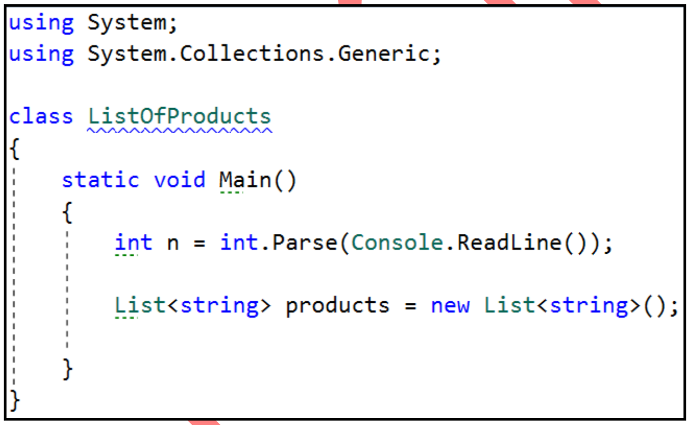


```
using System;

class ListOfProducts
{
    static void Main()
    {
        int n = int.Parse(Console.ReadLine());

    }
}
```

Then we need to create our **list of strings**, because the **products are strings**



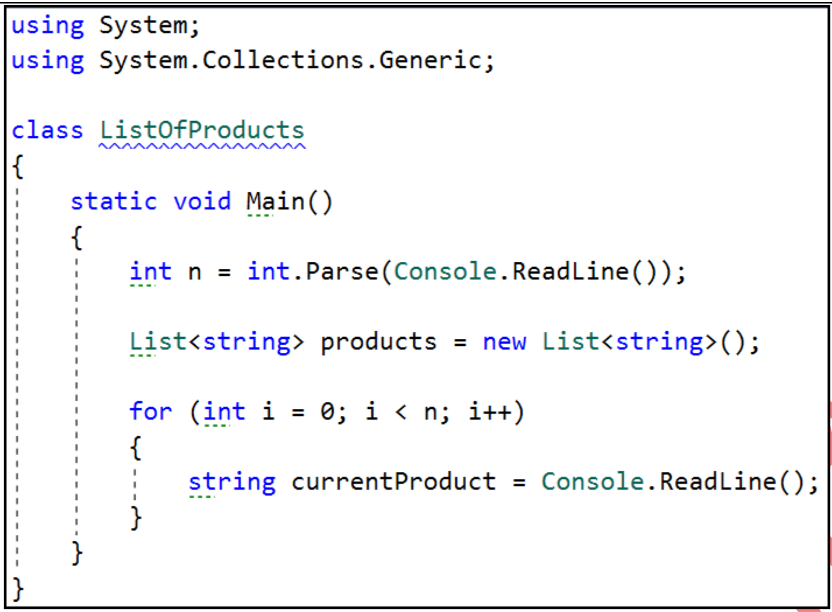
```
using System;
using System.Collections.Generic;

class ListOfProducts
{
    static void Main()
    {
        int n = int.Parse(Console.ReadLine());

        List<string> products = new List<string>();

    }
}
```

Then we need to iterate **n times** and **read our current product**



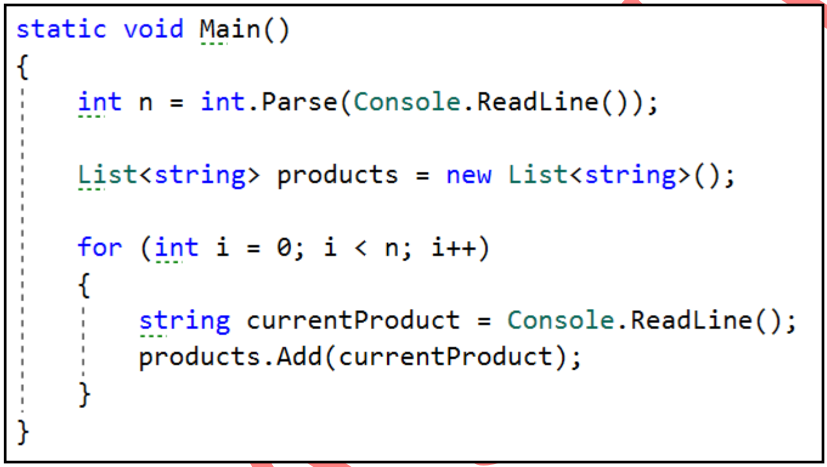
```
using System;
using System.Collections.Generic;

class ListOfProducts
{
    static void Main()
    {
        int n = int.Parse(Console.ReadLine());

        List<string> products = new List<string>();

        for (int i = 0; i < n; i++)
        {
            string currentProduct = Console.ReadLine();
        }
    }
}
```

The next step is to **add** the current product to the list

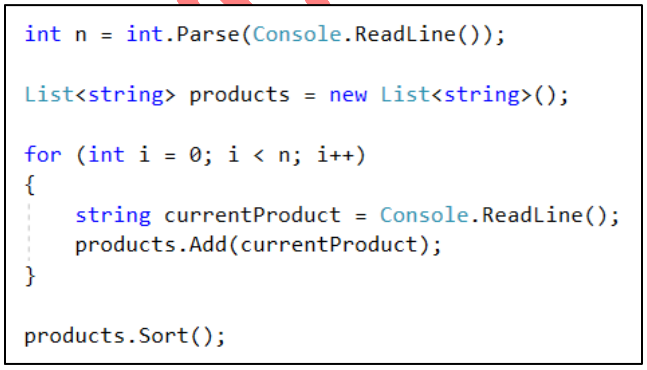


```
static void Main()
{
    int n = int.Parse(Console.ReadLine());

    List<string> products = new List<string>();

    for (int i = 0; i < n; i++)
    {
        string currentProduct = Console.ReadLine();
        products.Add(currentProduct);
    }
}
```

After we finish reading the products we **sort our list alphabetically**



```
int n = int.Parse(Console.ReadLine());

List<string> products = new List<string>();

for (int i = 0; i < n; i++)
{
    string currentProduct = Console.ReadLine();
    products.Add(currentProduct);
}

products.Sort();
```

- The **sort method** sorts the list in ascending order.

Finally, we have to **print our sorted** list. To do that we **loop through the list**.

```
for (int i = 0; i < products.Count; i++)
{
    Console.WriteLine($"{i + 1}.{products[i]}");
}
```

- We use **i + 1**, because we want to **start counting from 1**, we put the **'.'**, and **finally** we put **the actual product**

List Manipulation Basics

Write a program that reads a list of integers. Then until you receive **"end"**, you will be given different **commands:**

Add {number}: add a number to the end of the list

Remove {number}: remove number from the list

RemoveAt {index}: removes number at a given index

Insert {number} {index}: inserts a number at a given index

Note: All the indices will be valid!

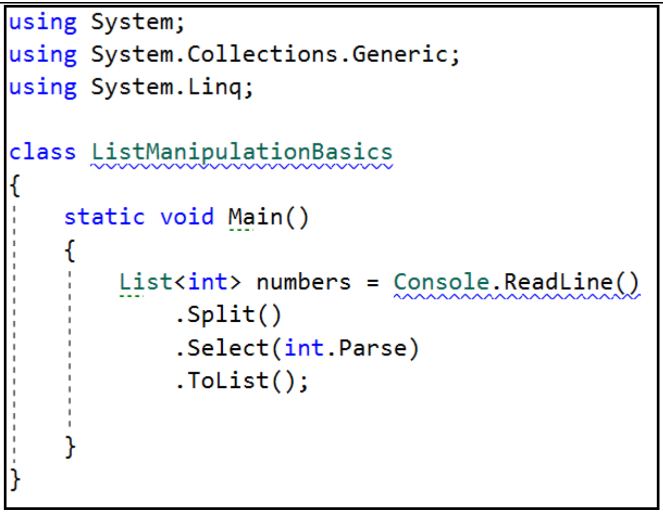
When you receive the **"end"** command print the **final state** of the list (**separated by spaces**)

Example

Input	Output
4 19 2 53 6 43 Add 3 Remove 2 RemoveAt 1 Insert 8 3 end	4 53 6 8 43 3

Solution

First let us read the list from the console.



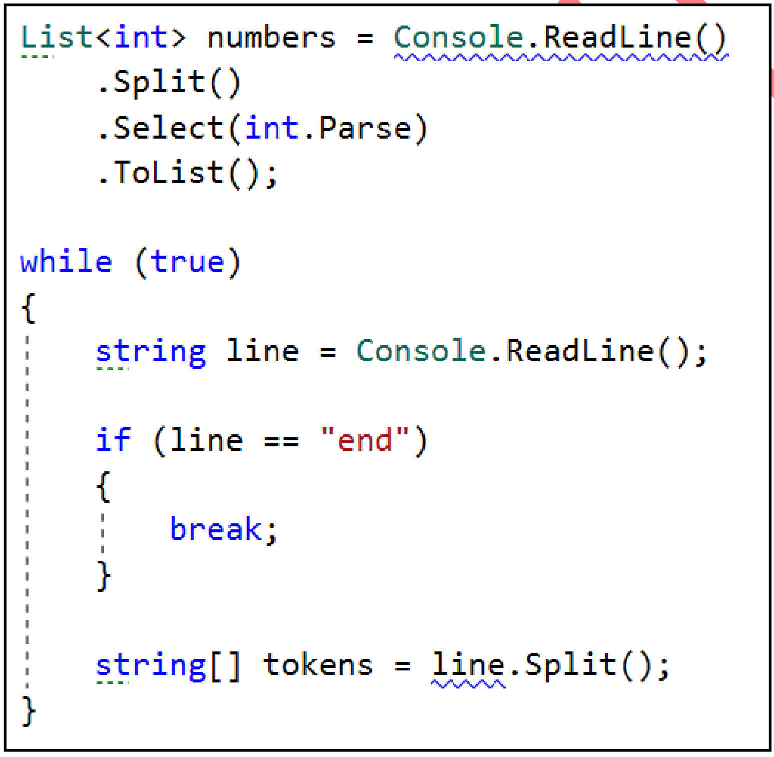
```
using System;
using System.Collections.Generic;
using System.Linq;

class ListManipulationBasics
{
    static void Main()
    {
        List<int> numbers = Console.ReadLine()
            .Split()
            .Select(int.Parse)
            .ToList();

    }
}
```

- We **split** the string read from the console, then we **loop through each element** and parse it to **int**
- This returns **IEnumarable<int>,** so we make it into a **list**

Next, we make the while loop for the commands and make switch statement for the commands



```
List<int> numbers = Console.ReadLine()
    .Split()
    .Select(int.Parse)
    .ToList();

while (true)
{
    string line = Console.ReadLine();

    if (line == "end")
    {
        break;
    }

    string[] tokens = line.Split();
}
```

- We break if the line is end, otherwise we split it into tokens

```
string[] tokens = line.Split();

switch (tokens[0])
{
    case "Add":
        break;
    case "Remove":
        break;
    case "RemoveAt":
        break;
    case "Insert":
        break;
}
```

Now let us implement each command

```
case "Add":
    int numberToAdd = int.Parse(tokens[1]);
    numbers.Add(numberToAdd);
    break;
case "Remove":
    int numberToRemove = int.Parse(tokens[1]);
    numbers.Remove(numberToRemove);
    break;
case "RemoveAt":
    int indexToRemove = int.Parse(tokens[1]);
    numbers.RemoveAt(indexToRemove);
    break;
case "Insert":
    int numberToInsert = int.Parse(tokens[1]);
    int indexToInsert = int.Parse(tokens[2]);
    numbers.Insert(indexToInsert, numberToInsert);
    break;
```

- For all commands **except from** the **"Insert", tokens[1]** is the **number/index**
- For the **"Insert"** command we receive a **number and an index** (**tokens[1], tokens[2]**)

Finally, we **print** the numbers, joined by **a single space**

```
Console.WriteLine(string.Join(" ", numbers));
```

List Manipulation Advanced

Now we will implement more complicated list commands, **extending the previous task**. Again, read a list, and until you receive **"end"** read commands:

Contains {number} – check if the list contains the number. If **yes** print **"Yes**", **otherwise** print **"No such number"**

PrintEven – print **all the numbers** that are **even separated by a space**

PrintOdd – print **all the numbers** that are **odd separated by a space**

GetSum – print the **sum of all the numbers**

Filter {condition} {number} – print all the numbers that **fulfill that condition**. The condition will be either **'<', '>', ">=", "<="**

After the **end command** print the list **only if** you have made some **changes** to the **original list**. **Changes** are made **only** from the commands from the **previous task**.

Example

Input	Output

`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`

Merging Lists

You are going to receive two lists with numbers. Create a result list which contains the numbers from both of the lists. The first element should be from the first list, the second from the second list and so on. If the length of the two lists are not equal, just add the remaining elements at the end of the list.

Example

Input	Output
3 5 2 43 12 3 54 10 23 76 5 34 2 4 12	3 76 5 5 2 34 43 2 12 4 3 12 54 10 23
76 5 34 2 4 12 3 5 2 43 12 3 54 10 23	76 3 5 5 34 2 2 43 4 12 12 3 54 10 23

Hint

- Read the two lists
- Create a result list
- Start looping through them until you reach the end of the smallest one
- Finally add the remaining elements (if any) to the end of the list