

C# programming exercises

In this pdf you can find C# programming exercises divided into different categories:

- [Basics](#)
- [Conditional statements](#)
- [Library functions](#)
- [LINQ](#)
- [Loops](#)
- [Miscellaneous](#)
- [Recursion](#)
- [Regular expressions](#)
- [Strings](#)

Tasks are at different level of difficulties. For each task you need to write a method with your solution. What's more, you also need to call your method with different input and check if returned result is OK.

For LINQ there is a different story. You don't need to write a separate method with solution, query is enough, but you have to test it to check if returns correct result.

You can prepare solutions in your favourite IDE ([e.g. Visual Studio](#)), or online compiler ([e.g. Ideone](#)).

GitHub: <https://github.com/piter0/Csharp-programming-exercises>

Basics

1. Add two numbers

Given three numbers, write a method that adds two first ones and multiplies them by a third one.

```
AddAndMultiply(2, 4, 5) → 30
```

[Go to solution](#)

2. Celsius to Fahrenheit

Given a temperature in Celsius degrees, write a method that converts it to Fahrenheit degrees. Remember that temperature below -273.15°C (absolute zero) does not exist!

```
CtoF(0) → "T = 32F"  
CtoF(100) → "T = 212F"  
CtoF(-300) → "Temperature below absolute zero!"
```

[Go to solution](#)

3. Elementary operations

Given two integers, write a method that returns results of their elementary arithmetic operations: addition, subtraction, multiplication, division. Remember that you can't divide any number by 0!

```
ElementaryOperations(3, 8) → 11, -5, 24, 0.375
```

[Go to solution](#)

4. Is result the same

Given two different arithmetic operations (addition, subtraction, multiplication, division), write a method that checks if they return the same result.

```
IsResultTheSame(2+2, 2*2) → true  
IsResultTheSame(9/3, 16-1) → false
```

[Go to solution](#)

5. Modulo operations

Given three integers, write a method that returns first number divided modulo by second one and these divided modulo by third one.

```
ModuloOperations(8, 5, 2) → 1
```

[Go to solution](#)

6. Swap two numbers

Given two integers, write a method that swaps them using temporary variable.

```
SwapTwoNumbers(87, 45) → "Before: a = 87, b = 45; After: a = 45, b = 87"  
SwapTwoNumbers(-13, 2) → "Before: a = -13, b = 2; After: a = 2, b = -13"
```

[Go to solution](#)

7. The cube of

Given a number, write a method that returns its cube.

```
TheCubeOf(2) → 8  
TheCubeOf(-5.5) → -166.375
```

[Go to solution](#)

Conditional statements

8. If sorted ascending

Given an array of three integers, write a method that checks if they are sorted in ascending order.

```
IfSortedAscending([3, 7, 10]) → true  
IfSortedAscending([74, 62, 99]) → false
```

[Go to solution](#)

9. If year is leap

Given a year as integer, write a method that checks if year is leap.

```
IfYearIsLeap(2016) → true  
IfYearIsLeap(2018) → false
```

[Go to solution](#)

10. If has neighbour

Given three letter long string, write a method that checks if at least one neighbour of middle letter is its neighbour in the alphabet.

```
IsLonelyIsland("XYZ") → true  
IsLonelyIsland("GWK") → false
```

[Go to solution](#)

11. Positive, negative or zero

Given a number, write a method that checks if it is positive, negative or zero.

```
PositiveNegativeOrZero(5.24) → positive  
PositiveNegativeOrZero(0.0) → zero  
PositiveNegativeOrZero(-994.53) → negative
```

[Go to solution](#)

12. If number contains 3

Write a method that checks if given number (positive integer) contains digit 3. Do not convert number to other type. Do not use built-in functions like `Contains()`, `StartsWith()`, etc.

```
IfNumberContains3(7201432) → true  
IfNumberContains3(87501) → false
```

[Go to solution](#)

13. Absolute value

Given an integer, write a method that returns its absolute value.

```
AbsoluteValue(6832) → 6832  
AbsoluteValue(-392) → 392
```

[Go to solution](#)

14. Divisible by 2 or 3

Given two integers, write a method that returns their multiplication if they are both divisible by 2 or 3, otherwise returns their sum.

```
DivisibleBy2Or3(15, 30) → 450  
DivisibleBy2Or3(2, 90) → 180  
DivisibleBy2Or3(7, 12) → 19
```

[Go to solution](#)

15. If consists of uppercase letters

Given a 3 characters long string, write a method that checks if it consists only of uppercase letters.

```
IfConsistsOfUppercaseLetters("xyz") → false  
IfConsistsOfUppercaseLetters("DOG") → true  
IfConsistsOfUppercaseLetters("L9#") → false
```

[Go to solution](#)

16. If greater than third one

Given an array of 3 integers, write a method that checks if multiplication or sum of two first numbers is greater than third one.

```
IfGreaterThanThirdOne([2, 7, 12]) → true  
IfGreaterThanThirdOne([-5, -8, 50]) → false
```

[Go to solution](#)

17. If number is even

Given an integer, write a method that checks if it is even.

```
IfNumberIsEven(721) → false  
IfNumberIsEven(1248) → true
```

[Go to solution](#)

Library functions

18. Negative or positive

Given a number, write a method that returns number to the power of 2 if negative or square root if positive or zero.

```
NegativeOrPositive(-2) → 4  
NegativeOrPositive(6.25) → 2.5
```

[Go to solution](#)

19. Replace x with y

Write a method that replaces every letter 'y' in the string with 'x'. Assume that string contains only lower case letters.

```
ReplaceXWithY("yellow") → "xellow"  
ReplaceXWithY("mushroom") → "mushroom"
```

[Go to solution](#)

20. To lower or to upper

Given a string which has at least two words separated by space, write a method that changes first word in the string to upper case, second to lower case and so on.

```
ToLowerOrToUpper("this is it") → "THIS is IT"
```

[Go to solution](#)

21. If starts with lower case

Given a string, write a method that checks if each word in the string starts with lower case and if so, removes this letter from the string.

```
IfStartsWithLowerCase("Alfa Beta gamma") → "Alfa Beta amma"
```

[Go to solution](#)

22. Greater number

Given two numbers, write a method that returns greater one.

```
GreaterNumber(2.1, 3) → 3  
GreaterNumber(-5, 0) → 0  
GreaterNumber(-111.22,111.222) → 111.222
```

[Go to solution](#)

LINQ

23. Transpose an array

Write a query that transposes square array (switch rows with columns).

```
[1,1,1,1    [1,2,3,4  
2,2,2,2    1,2,3,4  
3,3,3,3    → 1,2,3,4  
4,4,4,4]    1,2,3,4]
```

[Go to solution](#)

24. Unique values

Given a non-empty list of strings, return a list that contains only unique (non-duplicate) strings.

```
["abc", "xyz", "klm", "xyz", "abc", "abc", "rst"] → ["klm", "rst"]
```

[Go to solution](#)

25. Last word containing letter

Given a non-empty list of words, sort it alphabetically and return last word that contains letter 'e'.

```
["plane", "ferry", "car", "bike"] → "plane"
```

[Go to solution](#)

26. Numbers from range

Given an array of integers, write a query that returns list of numbers greater than 30 and less than 100.

```
[67, 92, 153, 15] → 67, 92
```

[Go to solution](#)

27. Minimum length

Write a query that returns words at least 5 characters long and make them uppercase.

```
"computer", "usb" → "COMPUTER"
```

[Go to solution](#)

28. Select words

Write a query that returns words starting with letter 'a' and ending with letter 'm'.

```
"mum", "amsterdam", "bloom" → "amsterdam"
```

[Go to solution](#)

29. Top 5 numbers

Write a query that returns top 5 numbers from the list of integers in descending order.

```
[78, -9, 0, 23, 54, 21, 7, 86] → 86 78 54 23 21
```

[Go to solution](#)

30. Frequency of letters

Write a query that returns letters and their frequencies in the string.

```
"gamma" → "Letter g occurs 1 time(s), Letter a occurs 2 time(s), Letter m  
occurs 2 time(s)"
```

[Go to solution](#)

31. Double letters

Write a query that returns double letters sequence in format: AA AB AC ... ZX ZY ZZ

```
(no input) → "AA AB AC ... AZ BA BB ... ZX ZY ZZ"
```

[Go to solution](#)

32. Shuffle an array

Write a query that shuffles sorted array.

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10] → [4, 9, 3, 5, 2, 10, 1, 6, 8, 7]  
[38, 24, 8, 0, -1, -17, -33, -100] → [0, -100, -17, 38, 8, -1, 24, -33,]
```

[Go to solution](#)

33. Decrypt number

Given a non-empty string consisting only of special chars (!, @, # etc.), return a number (as a string) where each digit corresponds to given special char on the keyboard (1 → !, 2 → @, 3 → # etc.).

```
"() ) (" → "9009"  
"*$(#&" → "84937"  
"!!!!!!!!!!" → "1111111111"
```

[Go to solution](#)

34. Replace substring

Write a query that replaces 'ea' substring with astersik (*) in given list of words.

```
"learn", "current", "deal" → "l*rn", "current", "d*l"
```

[Go to solution](#)

35. Most frequent character

Write a query that returns most frequent character in string. Assume that there is only one such character.

```
"panda" → 'a'  
"n093nfv034nie9" → 'n'
```

[Go to solution](#)

36. Square greater than 20

Write a query that returns list of numbers and their squares only if square is greater than 20

```
[7, 2, 30] → 7 - 49, 30 - 900
```

[Go to solution](#)

37. Uppercase only

Write a query that returns only uppercase words from string.

```
"DDD example CQRS Event Sourcing" → DDD, CQRS
```

[Go to solution](#)

38. Arrays dot product

Write a query that returns dot product of two arrays.

```
[1, 2, 3], [4, 5, 6] → 32  
[7, -9, 3, -5], [9, 1, 0, -4] → 74
```

[Go to solution](#)

39. Days names

Write a query that returns names of days.

```
daysNames → "Sunday Monday Tuesday Wednesday Thursday Friday Saturday"
```

[Go to solution](#)

Loops

40. Draw triangle

Write a method that draws triangle shape like below.

```
DrawTriangle() →  
  *  
  **  
  ***  
  ****  
  *****  
  *  
  **  
  ***  
  ****  
  *****  
  *  
  **  
  ***  
  ****  
  *****
```

[Go to solution](#)

41. To the power of

Given two integers, write a method that returns first number raised to the power of second one.

```
ToThePowerOf(-2, 3) → -8  
ToThePowerOf(5, 5) → 3125
```

[Go to solution](#)

42. The biggest number

Given an array of integers, write a method that returns the biggest number in this array.

```
TheBiggestNumber([190, 291, 145, 209, 280, 300]) → 291  
TheBiggestNumber([-9, -2, -7, -8, -4]) → -2
```

[Go to solution](#)

43. Two 7s next to each other

Given an array of positive digits, write a method that returns number of times that two 7's are next to each other in an array.

```
Two7sNextToEachOther([ 8, 2, 5, 7, 9, 0, 7, 7, 3, 1]) → 1  
Two7sNextToEachOther([ 9, 4, 5, 3, 7, 7, 7, 3, 2, 5, 7, 7 ]) → 3
```

[Go to solution](#)

44. Three increasing adjacent

Given an array of numbers, write a method that checks if there are three adjacent numbers where second is greater by 1 than the first one and third is greater by 1 than the second one.

```
ThreeIncreasingAdjacent([45, 23, 44, 68, 65, 70, 80, 81, 82 ]) → true  
ThreeIncreasingAdjacent([7, 3, 5, 8, 9, 3, 1, 4 ]) → false
```

[Go to solution](#)

45. Return even numbers

Write a method that returns a string of even numbers greater than 0 and less than 100.

```
ReturnEvenNumbers() → "2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38  
40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88  
90 92 94 96 98"
```

[Go to solution](#)

46. Sieve of Eratosthenes

Given an integer n ($n > 2$), write a method that returns prime numbers from range [2, n].

```
SieveOfEratosthenes(30) → [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

[Go to solution](#)

47. Sort array ascending

Given an array of integers, write a method that returns sorted array in ascending order.

```
SortArrayAscending([9, 5, 7, 2, 1, 8]) → [1, 2, 5, 7, 8, 9]
```

[Go to solution](#)

48. Sum and average

Given two integers n and m ($n \leq m$), write a method that returns sum of all integers and average from range [n, m].

```
SumAndAverage(11, 66) → "Sum: 2156, Average: 38.5"  
SumAndAverage(-10, 0) → "Sum: -55, Average: -5"
```

[Go to solution](#)

49. Sum double only

Given an array of objects, write a method that returns sum of objects of double type.

```
SumDoubleOnly(["abc", 5.6, 14, 'c', true, 'x', false, 567, 2.22]) → 7.82
```

[Go to solution](#)

50. Longest strictly increasing sequence

Given an array of integers, write a method that returns value of the longest strictly increasing sequence of numbers.

```
LongestStrictlyIncreasingSequence([0, 3, 4, 5, 6, 4, 9]) → 3  
LongestStrictlyIncreasingSequence([7, 7, 7, 7, 7]) → 0
```

[Go to solution](#)

51. Full sequence of letters

Given a string of two letters, where first one occurs before the second in the alphabet, write a method that returns full sequence of letters starting from first and ending at the second one.

```
FullSequenceOfLetters("ds") → "defghijklmnopqrs"  
FullSequenceOfLetters("or") → "opqr"
```

[Go to solution](#)

52. Digits sum

Given a non-negative number, write a method that returns sum of its digits.

```
DigitsSum(5434) → 16  
DigitsSum(904861) → 28
```

[Go to solution](#)

53. Digital root

Given a non-negative number, write a method that returns its digital root. From Wikipedia - digital root is a value obtained by an iterative process of summing digits, on each iteration using the result from the previous iteration to compute a digit sum. The process continues until a single-digit number is reached.

```
DigitalRoot(83) → 2  
DigitalRoot(40002938) → 8
```

[Go to solution](#)

54. Bits to number

Write a method that takes non-empty string of bits as an argument and returns number as integer.

```
BitsToNumber("1") → 1  
BitsToNumber("100010") → 34
```

[Go to solution](#)

55. Draw hourglass

Write a method that draws hourglass shape like below.

```
DrawHourglass() →
*****
*****
*****
*****
***
*
***
*****
*****
*****
*****
```

[Go to solution](#)

56. Draw parallelogram

Write a method that draws parallelogram shape like below.

```
DrawParallelogram() →
*****
*****
*****
*****
*****
```

[Go to solution](#)

57. Extract string

Given a string, write a method that returns substring from between two double hash signs (#).

```
ExtractString("##abc##def") → "abc"
ExtractString("12####78") → empty string
ExtractString("gar##d#en") → empty string
ExtractString("++##--##++") → "--"
```

[Go to solution](#)

58. Multiplication table

Write a method that prints 10 by 10 multiplication table. Remember about readability (spaces in the right place).

```
MultiplicationTable() →  
1  2  3  4  5  6  7  8  9 10  
2  4  6  8 10 12 14 16 18 20  
3  6  9 12 15 18 21 24 27 30  
4  8 12 16 20 24 28 32 36 40  
5 10 15 20 25 30 35 40 45 50  
6 12 18 24 30 36 42 48 54 60  
7 14 21 28 35 42 49 56 63 70  
8 16 24 32 40 48 56 64 72 80  
9 18 27 36 45 54 63 72 81 90  
10 20 30 40 50 60 70 80 90 100
```

[Go to solution](#)

59. Fractions sum

Given an integer n , write a method that returns sum of series $1 + (\frac{1}{2})^2 + (\frac{1}{3})^2 + \dots + (\frac{1}{n})^2$. Do not use library function!

```
FractionsSum(3) → 1.36111111111111  
FractionsSum(5) → 1.46361111111111
```

[Go to solution](#)

60. Letters balance

Given a string, write a method that checks if there are exactly the same letters on the left side and right side of the string. Assume string length is even and letters don't repeat on each side.

```
LettersBalance("fgvgvf") → true  
LettersBalance("lampsmpser") → false
```

[Go to solution](#)

61. Replace two words

Given a string in which two words are separated by a char, write a method that replaces these two words.

```
ReplaceWords("abc_xyz", '_') → xyz_abc  
ReplaceWords("trolling.master", '.') → master.trolling
```

[Go to solution](#)

62. Draw Christmas tree

Write a method that draws Christmas tree shape like below.

```
DrawChristmasTree() →  
  *  
 ***  
*****  
*****  
  *  
 ***  
*****  
*****  
  *  
 ***  
*****  
*****
```

[Go to solution](#)

Miscellaneous

63. How many days

Given two dates - first from the past and second as present date, write a method that returns numbers of days between these two dates.

```
HowManyDays((2006, 1, 31), Now) → 4652
```

[Go to solution](#)

Recursion

64. To the power of (recursion)

Given two integers, write a method that returns first number raised to the power of second number.

```
ToThePowerOfRecursion(2, 3) → 8  
ToThePowerOfRecursion(5, 2) → 25
```

[Go to solution](#)

65. Numbers multiplication

Given two integers a and b ($a \leq b$) as range, write a method that returns multiplication of numbers from given range.

```
NumbersMultiplication(5, 7) → 210  
NumbersMultiplication(50, 50) → 50
```

[Go to solution](#)

66. Digits multiplication

Given a positive integer, write a method that returns multiplication of all digits in the number.

```
DigitsMultiplication(456) → 120  
DigitsMultiplication(123) → 6
```

[Go to solution](#)

67. Factorial

Given a non-negative integer, write a method that returns factorial of a number.

```
Factorial(4) → 24  
Factorial(7) → 5040
```

[Go to solution](#)

68. Fibonacci number

Given a non-negative integer, write a method that returns n-th element of Fibonacci sequence.

```
FibonacciNumber(3) → 2  
FibonacciNumber(7) → 13
```

[Go to solution](#)

69. Is palindrome (recursion)

Given a string, write a method that checks if it is a palindrome. String length may be ≥ 0 .

```
IsPalindromeRecursion("xx") → true  
IsPalindromeRecursion("pendrive") → false
```

[Go to solution](#)

70. Minimum element

Given an array of integers and array's length, write a method that returns its minimum element.

```
MinimumElement([8, 5, 9], 3) → 5  
MinimumElement([-2, -9, 2, -3, 1, 0], 6) → -9
```

[Go to solution](#)

71. String in reverse order (recursion)

Given a string, write a method that prints it in reverse order.

```
StringInReverseOrderRecursion("abcde") → "edcba"  
StringInReverseOrderRecursion("Sed lectus est, elementum ut urna eu") → "ue  
anru tu mutnemele ,tse sutcel deS"
```

[Go to solution](#)

Regular expressions

72. Almost only letters

Given a string, write a method that checks if consists of letters only and ends with period. If string has more than one word, words are separated by space.

```
AlmostOnlyLetters("She is nice.") → true  
AlmostOnlyLetters("true 222.") → false
```

[Go to solution](#)

73. Check phone number

Given a phone number as a string, write a method that checks if it is in the format +XX YYY-YYY-YYY.

```
CheckPhoneNumber("+35 392-022-194") → true  
CheckPhoneNumber("+958 28492-503") → false
```

[Go to solution](#)

74. Decimal digit information

Given a string, write a method that checks if contains decimal digit and if yes returns its value and position.

```
DecimalDigitInformation("This is 9") → "Digit 9 at position 8"  
DecimalDigitInformation("ABCdef") → "No digit found!"
```

[Go to solution](#)

75. Every word in the string

Given a string, write a method that checks if every word begins with capital letter.

```
EveryWordInTheString("Use Of Technology") → true  
EveryWordInTheString("Rocket science") → false
```

[Go to solution](#)

76. Replace good with bad

Given a string, write a method that replaces every word 'good' with 'bad'. Assume that words to be replaced may consist of mixed cases (gOod, baD, etc.).

```
ReplaceGoodWithBad("g00d") → "good"  
ReplaceGoodWithBad("so b@d") → "so b@d"
```

[Go to solution](#)

Strings

77. Check brackets sequence

Given a sequence of brackets, write a method that checks if it has the same number of opening and closing brackets.

```
CheckBracketsSequence("((()))") → true  
CheckBracketsSequence("()()()") → false  
CheckBracketsSequence(")") → false
```

[Go to solution](#)

78. Add separator

Given a string and a separator, write a method that adds separator between each adjacent characters in a string.

```
AddSeparator("ABCD", "^") → "A^B^C^D^"  
AddSeparator("chocolate", "-") → "c-h-o-c-o-l-a-t-e"
```

[Go to solution](#)

79. Is palindrome

Given a string, write a method that checks if it is a palindrome (is read the same backward as forward). Assume that string may consist only of lower-case letters.

```
IsPalindrome("eye") → true  
IsPalindrome("home") → false
```

[Go to solution](#)

80. Length of string

Given a string, write a method that returns its length. Do not use library methods!

```
LengthOfString("computer") → 8  
LengthOfString("ice cream") → 9
```

[Go to solution](#)

81. Make uppercase

Given a string, write a method that returns new string in which every odd letter of the word is uppercase. String may consist of one or more words.

```
MakeUppercase("modem") → "MoDeM"  
MakeUppercase("BookWorm") → "BoOkWoRm"  
MakeUppercase("Aliquam dolor nisl?") → "AlIqUaM DoLoR NiSl?"
```

[Go to solution](#)

82. How many occurrences

Given a string and substring, write a method that returns number of occurrences of substring in the string. Assume that both are case-sensitive. You may need to use library function here.

```
HowManyOccurrences("do it now", "do") → 1  
HowManyOccurrences("empty", "d") → 0
```

[Go to solution](#)

83. Sort characters descending

Given a string, write a method that returns array of chars (ASCII characters) sorted in descending order.

```
SortCharactersDescending("onomatopoeia") → tpooooonmieaa  
SortCharactersDescending("fohjwf42os") → wsoojhff42
```

[Go to solution](#)

84. Revert words order

Given a string, write a method that returns new string with reverted words order. Pay attention to the punctuation at the end of the sentence (period).

```
RevertWordsOrder("John Doe.") → "Doe John."  
RevertWordsOrder("A, B. C") → "C B. A,"
```

[Go to solution](#)

85. Mix two strings

Given two strings, write a method that returns one string made of two strings. First letter of new string is first letter of first string, second letter of new string is first letter of second string and so on.

```
MixTwoStrings("aaa", "BBB") → "aBaBaB"  
MixTwoStrings("good one", "111") → "g1o1o1d one"
```

[Go to solution](#)

86. Number of words

Given a string, write a method that counts its number of words. Assume there are no leading and trailing whitespaces and there is only single whitespace between two consecutive words.

```
NumberOfWords("This is sample sentence") → 4  
NumberOfWords("OK") → 1
```

[Go to solution](#)

87. String in reverse order

Given a string, write a method that returns that string in reverse order.

```
StringInReverseOrder("qwerty") → "ytrewq"  
StringInReverseOrder("oe93 kr") → "rk 39eo"
```

[Go to solution](#)

88. Compress string

Given a non-empty string, write a method that returns it in compressed format.

```
CompressString("kkkktttrrrrrrrrrrr") → "k4t3r10"  
CompressString("p555ppp7www") → "p153p371w3"
```

[Go to solution](#)

89. Sum digits in string

Given a string, write a method which returns sum of all digits in that string. Assume that string contains only single digits.

```
SumDigitsInString("1q2w3e") → 6  
SumDigitsInString("L0r3m.1p5um") → 9  
SumDigitsInString("") → 0
```

[Go to solution](#)

Sample solutions

Add two numbers

```
using System;

namespace CSharpExercises.Exercises.Basics
{
    class AddAndMultiplyTask
    {
        public static double AddAndMultiply(double a, double b, double c)
        {
            return (a + b) * c;
        }

        public static void Main()
        {
            Console.WriteLine(AddAndMultiply(3, 6, 35));           // 315
            Console.WriteLine(AddAndMultiply(-12, 5, 17));         // -119
            Console.WriteLine(AddAndMultiply(-40, 50, 60));        // 600
            Console.WriteLine(AddAndMultiply(1.7, 9.9, 0.01));     // 0.116
        }
    }
}
```

Celsius to Fahrenheit

```
using System;

namespace CSharpExercises.Exercises.Basics
{
    class CtoFTask
    {
        public static string CtoF(double celsius)
        {
            double fahrenheit;

            if (celsius < -273.15)
            {
                return "Temperature below absolute zero!";
            }

            fahrenheit = celsius * 1.8 + 32;

            return $"T = {fahrenheit}F";
        }

        public static void Main()
        {
            Console.WriteLine(CtoF(0));      // T = 32F
            Console.WriteLine(CtoF(-300));   // Temperature below absolute zero!
            Console.WriteLine(CtoF(28.5));   // T = 83.3F
        }
    }
}
```

Elementary operations

```
using System;

namespace CSharpExercises.Exercises.Basics
{
    class ElementaryOperationsTask
    {
        public static string ElementaryOperations(int a, int b)
        {
            var addition = a + b;
            var subtraction = a - b;
            var multiplication = a * b;
            double division;

            if (b != 0)
            {
                division = a / (double)b;
            }
            else // assume that division by 0 returns 0
            {
                division = 0;
            }

            return String.Format($"a + b = {addition}, a - b = {subtraction}, a * b = {multiplication}, a / b = {division}");
        }

        public static void Main()
        {
            Console.WriteLine(ElementaryOperations(36, 15)); // a + b = 51, a - b = 21, a * b = 540, a / b = 2.4
            Console.WriteLine(ElementaryOperations(-375, 25)); // a + b = -350, a - b = -400, a * b = -9375, a / b = -15
        }
    }
}
```

Is result the same

```
using System;

namespace CSharpExercises.Exercises.Basics
{
    class IsResultTheSameTask
    {
        public static bool IsResultTheSame(double a, double b)
        {
            return a == b;
        }

        public static void Main()
        {
            Console.WriteLine(IsResultTheSame(3 * 3, 18 / 2));           // true
            Console.WriteLine(IsResultTheSame(3 + 7, 12 - 8));           // false
            Console.WriteLine(IsResultTheSame(3 * 7 * 8, 256 / 2 / 3)); // false
        }
    }
}
```


Modulo operations

```
using System;

namespace CSharpExercises.Exercises.Basics
{
    class ModuloOperationsTask
    {
        public static int ModuloOperations(int a, int b, int c)
        {
            return a % b % c;
        }

        public static void Main()
        {
            Console.WriteLine(ModuloOperations(542, 28, 7));    // 3
            Console.WriteLine(ModuloOperations(33, 10, 2));    // 1
            Console.WriteLine(ModuloOperations(2634, 892, 55)); // 25
        }
    }
}
```

Swap two numbers

```
using System;

namespace CSharpExercises.Exercises.Basics
{
    class SwapTwoNumbersTask
    {
        public static string SwapTwoNumbers(int a, int b)
        {
            var before = $"Before: a = {a}, b = {b}; ";
            int temp;

            temp = b;
            b = a;
            a = temp;

            string after = $"After: a = {a}, b = {b}";
            return before + after;
        }

        public static void Main()
        {
            Console.WriteLine(SwapTwoNumbers(23, 15));           // Before: a = 23, b =
15; After: a = 15, b = 23
            Console.WriteLine(SwapTwoNumbers(-123, 999));       // Before: a = -123, b
= 999; After: a = 999, b = -123
            Console.WriteLine(SwapTwoNumbers(0, 333));          // Before: a = 0, b =
333; After: a = 333, b = 0
        }
    }
}
```

The cube of

```
using System;

namespace CSharpExercises.Exercises.Basics
{
    class TheCubeOfTask
    {
        public static double TheCubeOf(double num)
        {
            return num * num * num;
        }

        public static void Main()
        {
            Console.WriteLine(TheCubeOf(15));    // 3375
            Console.WriteLine(TheCubeOf(0.25));  // 0.015625
            Console.WriteLine(TheCubeOf(-12));   // -1728
            Console.WriteLine(TheCubeOf(-0.1));  // -0.001
        }
    }
}
```

If sorted ascending

```
using System;

namespace CSharpExercises.Exercises.Conditional_statements
{
    class IfSortedAscendingTask
    {
        static bool IfSortedAscending(int[] arr)
        {
            return arr[0] <= arr[1] && arr[1] <= arr[2];
        }

        static void Main(string[] args)
        {
            Console.WriteLine(IfSortedAscending(new int[] { 3, 6, 9 })); // true
            Console.WriteLine(IfSortedAscending(new int[] { 34, 17, 90 })); //
false
            Console.WriteLine(IfSortedAscending(new int[] { -50, -24, -1 })); //
true
        }
    }
}
```

If year is leap

```
using System;

namespace CSharpExercises.Exercises.Conditional_statements
{
    class IfYearIsLeapTask
    {
        static bool IfYearIsLeap(int year)
        {
            return (year % 4 == 0 && year % 100 != 0) || year % 400 == 0;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(IfYearIsLeap(2020)); // true
            Console.WriteLine(IfYearIsLeap(1719)); // false
            Console.WriteLine(IfYearIsLeap(2000)); // true
            Console.WriteLine(IfYearIsLeap(1412)); // true
            Console.WriteLine(IfYearIsLeap(1582)); // false
        }
    }
}
```

If has neighbour

```
using System;

namespace CSharpExercises.Exercises.Conditional_statements
{
    class IfHasNeighbourTask
    {
        public static bool IfHasNeighbour(string word)
        {
            return word[0] == word[1] - 1 || word[0] == word[1] + 1 || word[2] ==
word[1] - 1 || word[2] == word[1] + 1;
        }

        public static void Main()
        {
            Console.WriteLine(IfHasNeighbour("DCA")); // true
            Console.WriteLine(IfHasNeighbour("PRT")); // false
        }
    }
}
```

Positive, negative or zero

```
using System;

namespace CSharpExercises.Exercises.Conditional_statements
{
    class PositiveNegativeOrZeroTask
    {
        static string PositiveNegativeOrZero(double num)
        {
            if (num > 0.0)
            {
                return "Positive";
            }
            else if (num < 0.0)
            {
                return "Negative";
            }

            return "Zero";
        }

        static void Main(string[] args)
        {
            Console.WriteLine(PositiveNegativeOrZero(3.14)); // Positive
            Console.WriteLine(PositiveNegativeOrZero(0.0)); // Zero
            Console.WriteLine(PositiveNegativeOrZero(-200.003)); // Negative
        }
    }
}
```

If number contains 3

```
using System;

namespace CSharpExercises.Exercises.Conditional_statements
{
    class IfNumberContains3Task
    {
        public static bool IfNumberContains3(int number)
        {
            while (number > 0)
            {
                if (number % 10 == 3)
                {
                    return true;
                }

                number /= 10;
            }

            return false;
        }

        public static void Main()
        {
            Console.WriteLine(IfNumberContains3(5384562)); // true
            Console.WriteLine(IfNumberContains3(0));        // false
            Console.WriteLine(IfNumberContains3(390462));    // true
        }
    }
}
```


Absolute value

```
using System;

namespace CSharpExercises.Exercises.Conditional_statements
{
    class AbsoluteValueTask
    {
        static int AbsoluteValue(int number)
        {
            return number >= 0 ? number : number * -1;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(AbsoluteValue(-690543)); // 690543
            Console.WriteLine(AbsoluteValue(2744));    // 2744
            Console.WriteLine(AbsoluteValue(0));        // 0
            Console.WriteLine(AbsoluteValue(-23));     // 23
        }
    }
}
```

Divisible by 2 or 3

```
using System;

namespace CSharpExercises.Exercises.Conditional_statements
{
    class DivisibleBy2Or3Task
    {
        static int DivisibleBy2Or3(int a, int b)
        {
            return (a % 2 == 0 && b % 2 == 0 || a % 3 == 0 && b % 3 == 0) ? a * b
: a + b;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(DivisibleBy2Or3(2, 18));    // 36
            Console.WriteLine(DivisibleBy2Or3(7, 0));     // 7
            Console.WriteLine(DivisibleBy2Or3(33, 9));     // 297
            Console.WriteLine(DivisibleBy2Or3(-72, 54));  // -3888
            Console.WriteLine(DivisibleBy2Or3(24, -80));  // -1920
            Console.WriteLine(DivisibleBy2Or3(444, 0));   // 0
        }
    }
}
```

If consists of uppercase letters

```
using System;

namespace CSharpExercises
{
    class Program
    {
        static bool IfConsistsOfUppercaseLetters(string str)
        {
            return (str[0] >= 65 && str[1] >= 65 && str[2] >= 65) && (str[0] <= 90
&& str[1] <= 90 && str[2] <= 90);
        }

        static void Main(string[] args)
        {
            Console.WriteLine(IfConsistsOfUppercaseLetters("drY")); // false
            Console.WriteLine(IfConsistsOfUppercaseLetters("LOL")); // true
            Console.WriteLine(IfConsistsOfUppercaseLetters("N0t")); // false
            Console.WriteLine(IfConsistsOfUppercaseLetters("$1r")); // false
        }
    }
}
```

If greater than third one

```
using System;

namespace CSharpExercises.Exercises.Conditional_statements
{
    class IfGreaterThanOrEqualToTask
    {
        static bool IfGreaterThanOrEqualTo(int[] arr)
        {
            return arr[0] + arr[1] > arr[2] || arr[0] * arr[1] > arr[2];
        }

        static void Main(string[] args)
        {
            Console.WriteLine(IfGreaterThanOrEqualTo(new int[] { 2, 8, 20 }));
            // false
            Console.WriteLine(IfGreaterThanOrEqualTo(new int[] { 10, 5, 22 }));
            // true
            Console.WriteLine(IfGreaterThanOrEqualTo(new int[] { -15, -25, 100 }));
            // true
            Console.WriteLine(IfGreaterThanOrEqualTo(new int[] { 11, 15, 166 }));
            // false
        }
    }
}
```

If number is even

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class IfNumberIsEvenTask
    {
        static bool IfNumberIsEven(int num)
        {
            return num % 2 == 0;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(IfNumberIsEven(8));           // true
            Console.WriteLine(IfNumberIsEven(54749));       // false
            Console.WriteLine(IfNumberIsEven(-43234670));   // true
            Console.WriteLine(IfNumberIsEven(0));           // true
            Console.WriteLine(IfNumberIsEven(-950541901)); // false
            Console.WriteLine(IfNumberIsEven(2140872324)); // true
        }
    }
}
```

Negative or positive

```
using System;

namespace CSharpExercises.Exercises.Library_functions
{
    class NegativeOrPositiveTask
    {
        public static double NegativeOrPositive(double num)
        {
            return num < 0 ? Math.Pow(num, 2) : Math.Sqrt(num);
        }

        public static void Main()
        {
            Console.WriteLine(NegativeOrPositive(72946));           //
270.085171751431
            Console.WriteLine(NegativeOrPositive(-4.726));         // 22.335076
            Console.WriteLine(NegativeOrPositive(0));              // 0
            Console.WriteLine(NegativeOrPositive(3.334));          //
1.82592442340859
            Console.WriteLine(NegativeOrPositive(-59));            // 3481
        }
    }
}
```

Replace x with y

```
using System;

namespace CSharpExercises.Exercises.Library_functions
{
    class ReplaceXWithYTask
    {
        public static string ReplaceXWithY(string word)
        {
            var words = word.Split(' ');

            for (var i = 0; i < words.Length; i++)
            {
                if (words[i].Contains("y"))
                {
                    words[i] = words[i].Replace("y", "x");
                }
            }

            return String.Join(" ", words);
        }

        public static void Main()
        {
            Console.WriteLine(ReplaceXWithY("yyy"));
            // xxx
            Console.WriteLine(ReplaceXWithY("strawberry youghurt"));
            // strawberrx xoughurt
            Console.WriteLine(ReplaceXWithY("tym ryhosx oifg 6 t6 ypeg ergh"));
            // txm rxhosx oifg 6 t6 xpeg ergh
            Console.WriteLine(ReplaceXWithY(""));
            // /empty string/
        }
    }
}
```

To lower or to upper

```
using System;

namespace CSharpExercises.Exercises.Library_functions
{
    class ToLowerOrToUpperTask
    {
        public static string ToLowerOrToUpper(string word)
        {
            var words = word.Split(' ');

            for (var i = 0; i < words.Length; i++)
            {
                words[i] = i % 2 == 0 ? words[i].ToUpper() : words[i].ToLower();
            }

            return String.Join(" ", words);
        }

        public static void Main()
        {
            Console.WriteLine(ToLowerOrToUpper("aaa BBB ccc DDD"));
            // AAA bbb CCC ddd
            Console.WriteLine(ToLowerOrToUpper("Etiam mollis lectus ac facilisis venenatis")); // ETIAM mollis LECTUS ac FACILISIS venenatis
            Console.WriteLine(ToLowerOrToUpper("th1s 15 5amp13 53nt3nc3"));
            // TH1S 15 5AMP13 53NT3NC3
        }
    }
}
```


If starts with lower case

```
using System;

namespace CSharpExercises.Exercises.Library_functions
{
    class IfStartsWithLowerCaseTask
    {
        public static string IfStartsWithLowerCase(string word)
        {
            var words = word.Split(' ');

            for (var i = 0; i < words.Length; i++)
            {
                if (Char.IsLower(words[i][0]))
                {
                    words[i] = words[i].Substring(1);
                }
            }

            return String.Join(" ", words);
        }

        public static void Main()
        {
            Console.WriteLine(IfStartsWithLowerCase("tthis iis ffake
ssentence.)); // this is fake sentence.
            Console.WriteLine(IfStartsWithLowerCase("Praesent vitae convallis
purus.)); // Praesent itae onvallis urus.
            Console.WriteLine(IfStartsWithLowerCase("1 2 3 7 8 9 a b c x y z"));
            // 1 2 3 7 8 9
        }
    }
}
```

Greater number

```
using System;

namespace CSharpExercises.Exercises.Library_functions
{
    class GreaterNumberTask
    {
        public static double GreaterNumber(double x, double y)
        {
            return Math.Max(x, y);
        }

        public static void Main()
        {
            Console.WriteLine(GreaterNumber(24.5, 127));    //127
            Console.WriteLine(GreaterNumber(-1391, -9123)); //-1391
            Console.WriteLine(GreaterNumber(-543, -543));    //543
        }
    }
}
```

Transpose an array

```
using System;
using System.Linq;

namespace CSharpExercises.Exercises.LINQ
{
    class TransposeAnArrayTask
    {
        public static void Main()
        {
            var array = new int[][] {new int[] { 1, 2, 3, 4, 5 },
                                     new int[] { 6, 7, 8, 9, 10 },
                                     new int[] { 11, 12, 13, 14, 15 },
                                     new int[] { 16, 17, 18, 19, 20 },
                                     new int[] { 21, 22, 23, 24, 25 }};

            var transposedArray = Enumerable.Range(0, array.Length).Select(x =>
array.Select(y => y[x]));

            foreach (var row in transposedArray)
            {
                foreach (var number in row)
                {
                    Console.Write(number + " ");
                }
                Console.WriteLine();
            }

            // 1 6 11 16 21
            // 2 7 12 17 22
            // 3 8 13 18 23
            // 4 9 14 19 24
            // 5 10 15 20 25
        }
    }
}
```

Unique values

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

namespace CSharpExercises.Exercises.LINQ
{
    class UniqueValuesTask
    {
        public static void Main()
        {
            var values = new List<string> { "Hi", "Meow", "Hello", "Meow", "Hi!",
            "Meow", "Hi", "Bye" };
            var uniqueValues = values
                .GroupBy(x => x)
                .Where(x => x.Count() == 1)
                .Select(x => x.Key)
                .ToList();

            foreach (var value in uniqueValues)
            {
                Console.WriteLine($"{value}"); // Hello Hi! Bye
            }
        }
    }
}
```

Last word containing letter

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

namespace CSharpExercises.Exercises.LINQ
{
    class LastWordContainingLetterTask
    {
        public static void Main()
        {
            var words = new List<string> { "cow", "dog", "elephant", "cat", "rat",
"squirrel", "snake", "stork" };

            var word = words.OrderBy(x => x)
                            .LastOrDefault(w => w.Contains("e"));

            Console.WriteLine($"{word}"); // squirrel
        }
    }
}
```

Numbers from range

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

namespace CSharpExercises.Exercises.LINQ
{
    class NumbersFromRangeTask
    {
        public static void Main()
        {
            var numbers = new List<int> { 30, 54, 3, 14, 25, 82, 1, 100, 23, 95 };

            var selectedNumbers = numbers.Where(x => x > 30).Where(x => x < 100);

            foreach (var selectedNumber in selectedNumbers)
            {
                Console.Write($"{selectedNumber} "); // 54 82 95
            }
        }
    }
}
```

Minimum length

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

namespace CSharpExercises.Exercises.LINQ
{
    class MinimumLengthTask
    {
        public static void Main()
        {
            var animals = new List<string> { "zebra", "elephant", "cat", "dog",
"rhino", "bat" };

            var selectedAnimals = animals.Where(s => s.Length >= 5).Select(x =>
x.ToUpper());

            foreach (var animal in selectedAnimals)
            {
                Console.Write($"{animal}, "); // ZEBRA, ELEPHANT, RHINO,
            }
        }
    }
}
```

Select words

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

namespace CSharpExercises.Exercises.LINQ
{
    class SelectWordsTask
    {
        public static void Main()
        {
            var words = new List<string> { "alabam", "am", "balalam", "tara", "",
            "a", "axeliam", "39yo0m", "trol" };

            var selectedWords = words.Where(s => s.StartsWith("a")).Where(s =>
            s.EndsWith("m"));

            foreach (var word in selectedWords)
            {
                Console.Write($"{word}, "); // alabam, am, axeliam,
            }
        }
    }
}
```


Top 5 numbers

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

namespace CSharpExercises.Exercises.LINQ
{
    class Top5NumbersTask
    {
        public static void Main()
        {
            var numbers = new List<int> { 6, 0, 999, 11, 443, 6, 1, 24, 54 };

            var top5 = numbers.OrderByDescending(x => x).Take(5);

            foreach (var number in top5)
            {
                Console.Write($"{number} "); // 999 443 54 24 11
            }
        }
    }
}
```

Frequency of letters

```
using System;
using System.Linq;

namespace CSharpExercises.Exercises.LINQ
{
    class FrequencyOfLettersTask
    {
        public static void Main()
        {
            var word = "abracadabra";

            var letters = word.GroupBy(x => x);

            foreach (var l in letters)
            {
                Console.WriteLine($"Letter {l.Key} occurs {l.Count()} time(s), ");
                // Letter a occurs 5 time(s), Letter b occurs 2 time(s), Letter r
occurs 2 time(s)
                // Letter r occurs 2 time(s), Letter c occurs 1 time(s), Letter d
occurs 1 time(s)
            }
        }
    }
}
```

Double letters

```
using System;
using System.Linq;

namespace CSharpExercises.Exercises.LINQ
{
    class DoubleLettersTask
    {
        public static void Main()
        {
            var doubleLetters = Enumerable.Range((char)65, 26)
                .SelectMany(x => Enumerable.Range((char)65, 26).Select(y
=> (char)x + "" + (char)y));

            foreach (var doubleLetter in doubleLetters)
            {
                Console.Write(doubleLetter + " "); // AA AB AC ... AZ BA BB ... ZX
ZY ZZ
            }
        }
    }
}
```

Shuffle an array

```
using System;
using System.Linq;

namespace CSharpExercises.Exercises.LINQ
{
    class ShuffleAnArrayTask
    {
        public static void Main()
        {
            var rnd = new Random();

            var array = new[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

            var shuffledArray = array.OrderBy(i => rnd.Next());

            foreach (var item in shuffledArray)
            {
                Console.Write(item + " "); // 4 10 3 6 2 8 1 9 7 5
            }
        }
    }
}
```

Decrypt number

```
using System;
using System.Linq;

namespace CSharpExercises.Exercises.LINQ
{
    class DecryptNumberTask
    {
        public static void Main()
        {
            var chars = new char[] { ')', '!', '@', '#', '$', '%', '^', '&', '*',
            '(' };

            var encryptedNumber = "#(@*%)$(&$*#&";

            var decryptedNumber = string.Join("", encryptedNumber.Select(c =>
            Array.IndexOf(chars, c)));

            Console.WriteLine(decryptedNumber); // 3928504974837
        }
    }
}
```

Replace substring

```
using System;
using System.Linq;

namespace CSharpExercises.Exercises.LINQ
{
    class ReplaceSubstringTask
    {
        public static void Main()
        {
            var words = new[] { "near", "speak", "tonight", "weapon", "customer",
"deal", "lawyer" };

            var modifiedWords = words.Select(w => w.Contains("ea") ?
w.Replace("ea", "*") : w);

            foreach(var word in modifiedWords)
            {
                Console.Write(word + " "); // n*r sp*k tonight w*pon customer d*l
lawyer
            }
        }
    }
}
```

Most frequent character

```
using System;
using System.Linq;

namespace CSharpExercises.Exercises.LINQ
{
    class MostFrequentCharacterTask
    {
        public static void Main()
        {
            var str = "49fjs492jfJs94KfoedK0iejksKdsj3";

            var mostFrequentCharacter = str.GroupBy(c => c).OrderByDescending(c =>
c.Count()).First().Key;

            Console.Write(mostFrequentCharacter);
        }
    }
}
```

Square greater than 20

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

namespace CSharpExercises.Exercises.LINQ
{
    class SquareGreaterThan20Task
    {
        public static void Main()
        {
            var numbers = new List<int> { 3, 9, 2, 4, 6, 5, 7 };

            var selectedNumbers = numbers.Where(x => x * x > 20);

            foreach (var s in selectedNumbers)
            {
                Console.Write($"{s} - {s * s}, "); // 9 - 81, 6 - 36, 5 - 25, 7 -
49,
            }
        }
    }
}
```


Uppercase only

```
using System;
using System.Linq;

namespace CSharpExercises.Exercises.LINQ
{
    class UppercaseOnlyTask
    {
        public static void Main()
        {
            var word = "THIS is UPPERCASE string LOL";

            var uppercaseOnly = word.Split(' ').Where(x => string.Equals(x,
x.ToUpper()));

            foreach (var u in uppercaseOnly)
            {
                Console.Write($"{u}, "); // THIS, UPPERCASE, LOL,
            }
        }
    }
}
```

Arrays dot product

```
using System;
using System.Linq;

namespace CSharpExercises.Exercises.LINQ
{
    class ArraysDotProduct
    {
        public static void Main()
        {
            var array1 = new int[] { 5, 8, 2, 9 };
            var array2 = new int[] { 1, 7, 2, 4 };

            var dotProduct = array1.Zip(array2, (a, b) => a * b).Sum();

            Console.WriteLine(dotProduct); // 101
        }
    }
}
```

Days names

```
using System;
using System.Linq;

namespace CSharpExercises.Exercises.LINQ
{
    class DaysNamesTask
    {
        public static void Main()
        {
            var daysNames = Enum.GetValues(typeof(DayOfWeek)).Cast<DayOfWeek>();

            foreach (var d in daysNames)
            {
                Console.Write($"{d} "); // Sunday Monday Tuesday Wednesday
                Console.WriteLine("Thursday Friday Saturday");
            }
        }
    }
}
```

Draw triangle

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class DrawTriangleTask
    {
        static void DrawTriangle()
        {
            for (var i = 0; i < 10; i++)
            {
                for (var j = 10; j > i; j--)
                {
                    Console.Write(" ");
                }
                for (var k = 10; k >= 10 - i; k--)
                {
                    Console.Write("*");
                }
                Console.WriteLine();
            }
        }

        static void Main(string[] args)
        {
            DrawTriangle();
        }
    }
}
```

To the power of

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class ToThePowerOfTask
    {
        static double ToThePowerOf(int b, int exp)
        {
            double result = 1;
            if (exp == 0)
            {
                return 1;
            }

            for (int i = 1; exp > 0 ? i <= exp : i <= exp * (-1); i++)
            {
                result *= b;
            }

            return exp > 0 ? result : 1 / result;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(ToThePowerOf(10, 0)); // 1
            Console.WriteLine(ToThePowerOf(5, -2)); // 0.04
            Console.WriteLine(ToThePowerOf(8, -8)); // 5.96046447753906E-08
            Console.WriteLine(ToThePowerOf(0, 5)); // 0
        }
    }
}
```

The biggest number

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class TheBiggestNumberTask
    {
        static int TheBiggestNumber(int[] numArr)
        {
            var theBiggest = numArr[0];

            for (var i = 1; i < numArr.Length; i++)
            {
                if (numArr[i] > theBiggest)
                {
                    theBiggest = numArr[i];
                }
            }

            return theBiggest;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(TheBiggestNumber(new int[] { 9, 4, 8, 1, 0, 2 }));
// 9
            Console.WriteLine(TheBiggestNumber(new int[] { -34, -54, -7, -40,
-123, -99 })); // -7
            Console.WriteLine(TheBiggestNumber(new int[] { 1009, 998, 1090, 3000,
2934, 4888 })); // 4888
        }
    }
}
```

Two 7s next to each other

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class Two7sNextToEachOtherTask
    {
        static int Two7sNextToEachOther(int[] arr)
        {
            var adjacent7s = 0;
            for (var i = 0; i < arr.Length - 1; i++)
            {
                if (arr[i] == 7 && arr[i + 1] == 7)
                {
                    adjacent7s++;
                }
            }

            return adjacent7s;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(Two7sNextToEachOther(new int[] { 7, 7, 8, 4, 3, 7,
2, 1, 0, 7 })); // 1
            Console.WriteLine(Two7sNextToEachOther(new int[] { 4, 7, 8, 2, 0, 5,
2, 7, 5, 8 })); // 0
            Console.WriteLine(Two7sNextToEachOther(new int[] { 7, 7, 7, 0, 2, 6,
4, 8, 6, 5, 2, 7, 7 })); // 3
        }
    }
}
```

Three increasing adjacent

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class ThreeIncreasingAdjacentTask
    {
        static bool ThreeIncreasingAdjacent(int[] arr)
        {
            var found = false;
            for (var i = 1; i <= arr.Length - 2; i++)
            {
                if (arr[i - 1] + 1 == arr[i] && arr[i + 1] - 1 == arr[i])
                {
                    found = true;
                }
            }

            return found;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(ThreeIncreasingAdjacent(new int[] { 7, 8, 9, 2, 4,
5, 0 })); // true
            Console.WriteLine(ThreeIncreasingAdjacent(new int[] { -9, 0, -1, -6,
-5, -4, -8, 0 })); // true
            Console.WriteLine(ThreeIncreasingAdjacent(new int[] { 15, 17, 14, 11,
18, 19, 16, 16 })); // false
        }
    }
}
```


Return even numbers

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class ReturnEvenNumbersTask
    {
        static string ReturnEvenNumbers()
        {
            var str = string.Empty;
            for (var i = 1; i < 100; i++)
            {
                if (i % 2 == 0)
                {
                    str += i + " ";
                }
            }

            return str;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(ReturnEvenNumbers()); // 2 4 6 8 10 12 14 16 18 20
22          // 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66
68          // 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98
        }
    }
}
```

Sieve of Eratosthenes

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class SieveOfEratosthenesTask
    {
        static bool[] SieveOfEratosthenes(int n)
        {
            var array = new bool[n];

            for (var i = 2; i < n; i++)
            {
                array[i] = true;
            }

            for (var j = 2; j * j <= n; j++)
            {
                if (array[j] == true)
                {
                    for (var k = j * j; k < n; k += j)
                    {
                        array[k] = false;
                    }
                }
            }
            return array;
        }

        static void Main(string[] args)
        {
            var arrayOfPrimes = SieveOfEratosthenes(100);
            for (var i = 0; i < arrayOfPrimes.Length; i++)
            {
                if (arrayOfPrimes[i] != false)
                {
                    Console.Write($"{i} "); // 2 3 5 7 11 13 17 19 23 29 31 37 41
                    43 47 53 59 61 67 71 73 79 83 89 97
                }
            }
        }
    }
}
```

Sort array ascending

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class SortArrayAscendingTask
    {
        static int[] SortArrayAscending(int[] arr)
        {
            int temp;
            for (var i = 0; i < arr.Length - 1; i++)
            {
                for (var j = 0; j < arr.Length - 1; j++)
                {
                    if (arr[j] > arr[j + 1])
                    {
                        temp = arr[j + 1];
                        arr[j + 1] = arr[j];
                        arr[j] = temp;
                    }
                }
            }

            return arr;
        }

        static void Main(string[] args)
        {
            var sortedArr = SortArrayAscending(new int[] { 0, -23, 9, 18, -51, 1,
90, 57, -1, 25 });

            foreach (var s in sortedArr)
            {
                Console.Write($"{s} "); // -51 -23 -1 0 1 9 18 25 57 90
            }
        }
    }
}
```

Sum and average

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class SumAndAverageTask
    {
        static string SumAndAverage(int lowest, int highest)
        {
            int sum = 0;
            int range = 0;
            double average = 0.0;
            for (var i = lowest; i <= highest; i++)
            {
                sum += i;
                range++;
            }

            average = sum / (double)range;

            return string.Format($"Sum: {sum}, Average: {average}");
        }

        static void Main(string[] args)
        {
            Console.WriteLine(SumAndAverage(20, 21)); // Sum: 41 Average: 20,5
            Console.WriteLine(SumAndAverage(55, 55)); // Sum: 55 Average: 55
            Console.WriteLine(SumAndAverage(0, 100)); // Sum: 5050 Average: 50
        }
    }
}
```

Sum double only

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class SumDoubleOnlyTask
    {
        static double SumDoubleOnly(object[] obj)
        {
            double sum = 0.0;
            for (var i = 0; i < obj.Length; i++)
            {
                if (obj[i] is double)
                {
                    sum += (double)obj[i];
                }
            }

            return sum;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(SumDoubleOnly(new object[] { 8.9, "dog", 6, 'c',
null, 15.99, 745, true })); // 24.89
        }
    }
}
```

Longest strictly increasing sequence

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class LongestStrictlyIncreasingSequenceTask
    {
        static int LongestStrictlyIncreasingSequence(int[] array)
        {
            int tempLongest = 0;
            int longest = 0;

            for (var i = 0; i < array.Length - 1; i++)
            {
                if (array[i + 1] > array[i])
                {
                    tempLongest++;
                }
                else
                {
                    tempLongest = 0;
                }

                if (tempLongest > longest)
                {
                    longest = tempLongest;
                }
            }

            return longest;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(LongestStrictlyIncreasingSequence(new int[] { 4, 7,
2, 6, 4, 5, 6, 7, 8, 0, 7, 1, 2, 3 })); // 4
            Console.WriteLine(LongestStrictlyIncreasingSequence(new int[] { 1, 0,
1, 0, 1, 0, 1, 0, 1, 0 })); // 1
            Console.WriteLine(LongestStrictlyIncreasingSequence(new int[] { 2, 3,
4, 5, 6, 7, 8 })); // 6
            Console.WriteLine(LongestStrictlyIncreasingSequence(new int[] { 1, 1,
1, 1, 1, })); // 0
        }
    }
}
```

Full sequence of letters

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class FullSequenceOfLettersTask
    {
        static string FullSequenceOfLetters(string word)
        {
            var fullSequence = string.Empty;
            for (var i = word[0], j = 0; i <= word[1]; i++, j++)
            {
                fullSequence += (char)(word[0] + j);
            }

            return fullSequence;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(FullSequenceOfLetters("bg")); // bcdefg
            Console.WriteLine(FullSequenceOfLetters("xy")); // xy
            Console.WriteLine(FullSequenceOfLetters("az")); //
abcdefghijklmnopqrstuvwxyz
        }
    }
}
```

Digits sum

```
using System;

namespace CSharpExercises
{
    class DigitsSumTask
    {
        public static int DigitsSum(uint number)
        {
            var sum = 0;
            var i = 10;
            var j = 1;

            while (number / j >= 1)
            {
                sum += (int)(number % i / j);

                i *= 10;
                j *= 10;
            }

            return sum;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(DigitsSum(5));           // 5
            Console.WriteLine(DigitsSum(1029584739)); // 48
            Console.WriteLine(DigitsSum(999999999));  // 72
        }
    }
}
```


Digital root

```
namespace CSharpExercises.Exercises.Loops
{
    class DigitalRootTask
    {
        public static int DigitalRoot(uint number)
        {
            while (number / 10 != 0)
            {
                uint sum = 0;
                var i = 10;
                var j = 1;

                while (number / j >= 1)
                {
                    sum += (uint)(number % i / j);

                    i *= 10;
                    j *= 10;
                }

                number = sum;
            }

            return (int)number;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(DigitalRoot(5));           // 5
            Console.WriteLine(DigitalRoot(1029584739)); // 3
            Console.WriteLine(DigitalRoot(999999999));  // 9
        }
    }
}
```

Bits to number

```
using System;

namespace CSharpExercises
{
    class BitsToNumberTask
    {
        static int BitsToNumber(string bits)
        {
            var number = 0;

            for (var i = 0; i < bits.Length; i++)
            {
                number += (int)(char.GetNumericValue(bits[i]) * Math.Pow(2,
bits.Length - i - 1));
            }

            return number;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(BitsToNumber("0"));           // 0
            Console.WriteLine(BitsToNumber("00001011000001")); // 705
            Console.WriteLine(BitsToNumber("10001110001010100")); // 72288
        }
    }
}
```

Draw hourglass

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class DrawHourglassTask
    {
        static void DrawHourglass()
        {
            for (var i = 0; i <= 10; i++)
            {
                for (var j = 0; j < (i <= 5 ? i : 10 - i); j++)
                {
                    Console.Write(" ");
                }
                for (var k = i <= 5 ? i : 10 - i; k <= (i <= 5 ? 10 - i : i); k++)
                {
                    Console.Write("*");
                }
                for (var m = 10 - i; m < 10; m++)
                {
                    Console.Write(" ");
                }
                Console.WriteLine();
            }
        }

        static void Main(string[] args)
        {
            DrawHourglass();
        }
    }
}
```

Draw parallelogram

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class DrawParallelogramTask
    {
        static void DrawParallelogram()
        {
            for (var j = 0; j < 5; j++)
            {
                for (var k = 0; k < 5 - j; k++)
                {
                    Console.Write(" ");
                }
                for (var m = 0; m < 15; m++)
                {
                    Console.Write("*");
                }
                Console.WriteLine();
            }
        }

        static void Main(string[] args)
        {
            DrawParallelogram();
        }
    }
}
```

Extract string

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class ExtractStringTask
    {
        static string ExtractString(string word)
        {
            var extractedWord = string.empty;
            var firstOccurrence = false;
            var secondOccurrence = false;

            for (var i = 0; i <= word.Length - 1; i++)
            {
                if (word[i] == '#' && word[i + 1] == '#')
                {
                    firstOccurrence = true;
                    for (var j = i + 2; j <= word.Length - 1; j++)
                    {
                        if (word[j] == '#' && word[j + 1] == '#')
                        {
                            secondOccurrence = true;
                            return extractedWord;
                        }
                        extractedWord += word[j];
                    }
                }
            }

            return string.empty;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(ExtractString("kFp##jFoRj##pL")); // jFoRj
            Console.WriteLine(ExtractString("abc##def"));         // /empty string/
            Console.WriteLine(ExtractString("##123456789##"));     // 123456789
            Console.WriteLine(ExtractString("no####thing"));      // /empty string/
            Console.WriteLine(ExtractString("empty"));             // /empty string/
        }
    }
}
```

Multiplication table

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class MultiplicationTableTask
    {
        static void MultiplicationTable()
        {
            for (var i = 1; i <= 10; i++)
            {
                for (var j = 1; j <= 10; j++)
                {
                    if (i == 1)
                    {
                        if (i * j < 10)
                        {
                            Console.Write($" {i * j} ");
                        }
                        else
                        {
                            Console.Write($"{i * j} ");
                        }
                    }
                    else if (i > 1 && i < 10)
                    {
                        if (i * j < 10)
                        {
                            Console.Write($" {i * j} ");
                        }
                        else
                        {
                            Console.Write($"{i * j} ");
                        }
                    }
                    else
                    {
                        Console.Write($"{i * j} ");
                    }
                }
                Console.WriteLine();
            }
        }

        static void Main(string[] args)
        {
            MultiplicationTable();
            // 1 2 3 4 5 6 7 8 9 10
            // 2 4 6 8 10 12 14 16 18 20
            // 3 6 9 12 15 18 21 24 27 30
        }
    }
}
```

```
        // 4  8 12 16 20 24 28 32 36 40
        // 5 10 15 20 25 30 35 40 45 50
        // 6 12 18 24 30 36 42 48 54 60
        // 7 14 21 28 35 42 49 56 63 70
        // 8 16 24 32 40 48 56 64 72 80
        // 9 18 27 36 45 54 63 72 81 90
        //10 20 30 40 50 60 70 80 90 100
    }
}
}
```

Fractions sum

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class FractionsSumTask
    {
        static double FractionsSum(int num)
        {
            double sum = 0.0;
            for (var i = 1; i <= num; i++)
            {
                sum += (1 / (double)(i * i));
            }

            return sum;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(FractionsSum(2)); // 1.25
            Console.WriteLine(FractionsSum(7)); // 1.5117970521542
            Console.WriteLine(FractionsSum(10)); // 1.54976773116654
        }
    }
}
```


Letters balance

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class LettersBalanceTask
    {
        static bool LettersBalance(string word)
        {
            bool isBalanced;
            for (var i = 0; i < word.Length / 2; i++)
            {
                isBalanced = false;
                for (var j = word.Length - 1; j >= word.Length / 2; j--)
                {
                    if (word[i] == word[j])
                    {
                        isBalanced = true;
                    }
                }

                if (!isBalanced)
                {
                    return false;
                }
            }

            return true;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(LettersBalance("kfdfdk"));           // true
            Console.WriteLine(LettersBalance("reyjer"));           // false
            Console.WriteLine(LettersBalance("wkxzcsazsckawx"));   // true
            Console.WriteLine(LettersBalance("pkmqsdedeskgqm"));   // false
        }
    }
}
```

Replace two words

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class ReplaceWordsTask
    {
        public static string ReplaceWords(string word, char ch)
        {
            var firstWord = string.Empty;
            var secondWord = string.Empty;

            for (var i = 0; i <= word.Length - 1; i++)
            {
                if (word[i] != ch)
                {
                    secondWord += word[i];
                }
                else
                {
                    for (var j = i + 1; j <= word.Length - 1; j++)
                    {
                        firstWord += word[j];
                    }
                    break;
                }
            }

            return firstWord + ch + secondWord;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(ReplaceWords("dog_octopus", '_'));    //
octopus_dog
            Console.WriteLine(ReplaceWords("a.b", '.'));            //
b.a
            Console.WriteLine(ReplaceWords("good very", ' '));      // very
good
        }
    }
}
```

Draw Christmas tree

```
using System;

namespace CSharpExercises.Exercises.Loops
{
    class DrawChristmasTreeTask
    {
        static void DrawChristmasTree()
        {
            for (var i = 0; i < 3; i++)
            {
                for (var j = 0; j < 7; j += 2)
                {
                    for (var k = 0; k < (7 - j) / 2; k++)
                    {
                        Console.Write(" ");
                    }
                    for (var m = 0; m <= j; m++)
                    {
                        Console.Write("*");
                    }
                    for (var n = (7 - j) / 2; n < 7; n++)
                    {
                        Console.Write(" ");
                    }
                    Console.WriteLine();
                }
            }
        }

        static void Main(string[] args)
        {
            DrawChristmasTree();
        }
    }
}
```

How many days

```
using System;

namespace CSharpExercises.Exercises.Miscellaneous
{
    class HowManyDaysTask
    {
        public static int HowManyDays(DateTime dateInThePast, DateTime dateNow)
        {
            return (dateNow - dateInThePast).Days;
        }

        public static void Main()
        {
            Console.WriteLine(HowManyDays(new DateTime(2005, 12, 8),
DateTime.Now)); //4706
        }
    }
}
```

To the power of (recursion)

```
using System;

namespace CSharpExercises.Exercises.Recursion
{
    class ToThePowerOfRecursionTask
    {
        static int ToThePowerOfRecursion(int b, int exp)
        {
            return exp == 0 ? 1 : exp > 1 ? b * ToThePowerOfRecursion(b, exp - 1)
: b;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(ToThePowerOfRecursion(10, 0)); // 1
            Console.WriteLine(ToThePowerOfRecursion(3, 7));  // 2187
            Console.WriteLine(ToThePowerOfRecursion(2, 10)); // 1024
        }
    }
}
```

Numbers multiplication

```
using System;

namespace CSharpExercises.Exercises.Recursion
{
    class NumbersMultiplicationTask
    {
        static int NumbersMultiplication(int from, int to)
        {
            while (from == to)
            {
                return from;
            }

            return from * NumbersMultiplication(from + 1, to);
        }

        static void Main(string[] args)
        {
            Console.WriteLine($"{NumbersMultiplication(1, 5)}"); // 120
            Console.WriteLine($"{NumbersMultiplication(-27, -22)}"); // 213127200
            Console.WriteLine($"{NumbersMultiplication(44, 44)}"); // 44
        }
    }
}
```

Digits multiplication

```
using System;

namespace CSharpExercises.Exercises.Recursion
{
    class DigitsMultiplicationTask
    {
        static int DigitsMultiplication(int num)
        {
            return num > 10 ? num % 10 * DigitsMultiplication(num / 10) : num %
10;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(DigitsMultiplication(1234));           // 24
            Console.WriteLine(DigitsMultiplication(94632));         // 1296
            Console.WriteLine(DigitsMultiplication(222222222));     // 512
        }
    }
}
```

Factorial

```
using System;

namespace CSharpExercises.Exercises.Recursion
{
    class FactorialTask
    {
        static long Factorial(int num)
        {
            return num == 0 || num == 1 ? 1 : num * Factorial(num - 1);
        }

        static void Main(string[] args)
        {
            Console.WriteLine(Factorial(5)); // 120
            Console.WriteLine(Factorial(1)); // 1
            Console.WriteLine(Factorial(14)); // 87178291200
            Console.WriteLine(Factorial(0)); // 1
            Console.WriteLine(Factorial(20)); // 2432902008176640000
        }
    }
}
```


Fibonacci number

```
using System;

namespace CSharpExercises.Exercises.Recursion
{
    class FibonacciNumberTask
    {
        static int FibonacciNumber(int num)
        {
            return num <= 1 ? num : FibonacciNumber(num - 2) + FibonacciNumber(num
- 1);
        }

        static void Main(string[] args)
        {
            Console.WriteLine(FibonacciNumber(10)); // 55
            Console.WriteLine(FibonacciNumber(0)); // 0
            Console.WriteLine(FibonacciNumber(20)); // 6765
            Console.WriteLine(FibonacciNumber(13)); // 233
        }
    }
}
```

Is palindrome (recursion)

```
using System;

namespace CSharpExercises.Exercises.Recursion
{
    class IsPalindromeRecursionTask
    {
        static bool IsPalindromeRecursion(string str)
        {
            if (str.Length == 1 || (str.Length == 2 && str[0] == str[1]))
            {
                return true;
            }
            else if (str.Length > 1)
            {
                if (str[0] != str[str.Length - 1])
                {
                    return false;
                }

                return IsPalindromeRecursion(str.Substring(1, str.Length - 2));
            }

            return false;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(IsPalindromeRecursion("aa"));           // true
            Console.WriteLine(IsPalindromeRecursion("dad"));         // true
            Console.WriteLine(IsPalindromeRecursion("apple"));       // false
            Console.WriteLine(IsPalindromeRecursion("deleveled"));   // true
            Console.WriteLine(IsPalindromeRecursion(""));            // false
            Console.WriteLine(IsPalindromeRecursion("hannah"));     // true
            Console.WriteLine(IsPalindromeRecursion("X"));           // true
        }
    }
}
```

Minimum element

```
using System;

namespace CSharpExercises.Exercises.Recursion
{
    class MinimumElementTask
    {
        static int MinimumElement(int[] arr, int size)
        {
            if (size == 1)
            {
                return arr[0];
            }

            return arr[size - 1] < MinimumElement(arr, size - 1) ? arr[size - 1] :
MinimumElement(arr, size - 1);
        }

        static void Main(string[] args)
        {
            Console.WriteLine(MinimumElement(new int[] { 7, 2, 9, 5, 4 }, 5));
// 2
            Console.WriteLine(MinimumElement(new int[] { -45, -6, 39, 96, -20, 0,
-100 }, 7)); // -100
            Console.WriteLine(MinimumElement(new int[] { 830, 905, 999, 831, 920,
900 }, 6)); // 830
        }
    }
}
```

String in reverse order (recursion)

```
using System;

namespace CSharpExercises.Exercises.Recursion
{
    class StringInReverseOrderRecursionTask
    {
        static string StringInReverseOrderRecursion(string str)
        {
            return str.Length > 0 ? str[str.Length - 1] +
StringInReverseOrderRecursion(str.Substring(0, str.Length - 1)) : str;
        }

        static void Main(string[] args)
        {
            var str1 = "A";
            var str2 = "34 ( 9 9@";
            var str3 = "eMpIrE";
            var str4 = string.Empty;

            Console.WriteLine(StringInReverseOrderRecursion(str1)); // A
            Console.WriteLine(StringInReverseOrderRecursion(str2)); // *@9 9 ( 43
            Console.WriteLine(StringInReverseOrderRecursion(str3)); // ErIpMe
            Console.WriteLine(StringInReverseOrderRecursion(str4)); //

        }
    }
}
```

Almost only letters

```
using System;
using System.Text.RegularExpressions;

namespace CSharpExercises.Exercises.Regular_expressions
{
    class AlmostOnlyLettersTask
    {
        public static bool AlmostOnlyLetters(string word)
        {
            var regex = new Regex(@"^[A-Za-z\s]+\.$");
            var match = regex.Match(word);

            return match.Success;
        }

        public static void Main()
        {
            Console.WriteLine(AlmostOnlyLetters("Very hot.")); // true
            Console.WriteLine(AlmostOnlyLetters("full of hope")); // false
            Console.WriteLine(AlmostOnlyLetters("")); // false
            Console.WriteLine(AlmostOnlyLetters("short.")); // true
        }
    }
}
```

Check phone number

```
using System;
using System.Text.RegularExpressions;

namespace CSharpExercises.Exercises.Regular_expressions
{
    class CheckPhoneNumberTask
    {
        public static bool CheckPhoneNumber(string phoneNumber)
        {
            var regex = new Regex(@"^\+\d{2}\s(\d{3}\-){2}(\d{3})");
            var match = regex.Match(phoneNumber);

            return match.Success;
        }

        public static void Main()
        {
            Console.WriteLine(CheckPhoneNumber("+48 592-045-990")); // true
            Console.WriteLine(CheckPhoneNumber("+999 543-000-305")); // false
            Console.WriteLine(CheckPhoneNumber("00 204-145-722")); // false
            Console.WriteLine(CheckPhoneNumber("+47 420-053-934")); // true
        }
    }
}
```

Decimal digit information

```
using System;
using System.Text.RegularExpressions;

namespace CSharpExercises.Exercises.Regular_expressions
{
    class DecimalDigitInformationTask
    {
        public static string DecimalDigitInformation(string word)
        {
            var regex = new Regex(@"\d");
            var match = regex.Match(word);

            return match.Success ? $"Digit {match.Value} at position
{match.Index}" : $"No digit found!";
        }

        public static void Main()
        {
            Console.WriteLine(DecimalDigitInformation("The 7 is the digit!"));
            // Digit 7 at position 4
            Console.WriteLine(DecimalDigitInformation("Hamster and lion"));
            // No digit found!
            Console.WriteLine(DecimalDigitInformation("1st"));           // Digit
1 at position 0
        }
    }
}
```

Every word in the string

```
using System;
using System.Text.RegularExpressions;

namespace CSharpExercises.Exercises.Regular_expressions
{
    class EveryWordInTheStringTask
    {
        public static bool EveryWordInTheString(string word)
        {
            var regex = new Regex(@"^([A-Z]\w*\s*)+\W*$");
            var match = regex.Match(word);

            return match.Success;
        }

        public static void Main()
        {
            Console.WriteLine(EveryWordInTheString("I Love You"));
            // true
            Console.WriteLine(EveryWordInTheString("Greater Than 9"));
            // false
            Console.WriteLine(EveryWordInTheString("Pig And Horse!!!"));
            // true
            Console.WriteLine(EveryWordInTheString("Make    Some
Whitespaces?")); // true
            Console.WriteLine(EveryWordInTheString("As Fit As a Fiddle."));
            // false
        }
    }
}
```


Replace good with bad

```
using System;
using System.Text.RegularExpressions;

namespace CSharpExercises.Exercises.Regular_expressions
{
    class ReplaceGoodWithBadTask
    {
        public static string ReplaceGoodWithBad(string word)
        {
            var output = string.Empty;
            return output = Regex.Replace(word, @"((G|g)(O|o){2}(D|d))", "bad");
        }

        public static void Main()
        {
            Console.WriteLine(ReplaceGoodWithBad("Very GoOd"));           // Very
bad                               Console.WriteLine(ReplaceGoodWithBad("GooDgOOdG00Dgood")); //
badbadbadbad                     Console.WriteLine(ReplaceGoodWithBad("Not so g00d"));       // Not so
g00d
        }
    }
}
```

Check brackets sequence

```
using System;

namespace CSharpExercises.Exercises.Strings
{
    class CheckBracketsSequenceTask
    {
        static bool CheckBracketsSequence(string sequence)
        {
            var check = 0;

            for (int i = 0; i < sequence.Length; i++)
            {
                check = sequence[i] == '(' ? ++check : --check;
            }

            return check == 0;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(CheckBracketsSequence("(((())()))")); //
            true
            Console.WriteLine(CheckBracketsSequence(""))); //
            false
            Console.WriteLine(CheckBracketsSequence(")()()(")); //
            true
            Console.WriteLine(CheckBracketsSequence("()()()()")); //
            false
            Console.WriteLine(CheckBracketsSequence("(((()(((())()))))()()()")); //
            true
        }
    }
}
```

Add separator

```
using System;

namespace CSharpExercises.Exercises.Strings
{
    class AddSeparatorTask
    {
        static string AddSeparator(string word, string separator)
        {
            var separatedWord = string.Empty;

            for (var i = 0; i < word.Length; i++)
            {
                separatedWord += i < word.Length - 1 ? word[i] + separator :
word[i].ToString();
            }

            return separatedWord;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(AddSeparator("computer", "_")); // c_o_m_p_u_t_e_r
            Console.WriteLine(AddSeparator("$*(#", " "));      // $ * ( #
            Console.WriteLine(AddSeparator("AC", "B"));        // ABC
            Console.WriteLine(AddSeparator("octopus", "X"));   // oXcXtXoXpXuXs
        }
    }
}
```

Is palindrome

```
using System;

namespace CSharpExercises.Strings
{
    class IsPalindromeTask
    {
        static bool IsPalindrome(string str)
        {
            for (var i = 0; i < str.Length / 2; i++)
            {
                if (str[i] != str[str.Length - 1 - i])
                {
                    return false;
                }
            }

            return true;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(IsPalindrome("madam")); //
true           Console.WriteLine(IsPalindrome("123454321")); //
true           Console.WriteLine(IsPalindrome("apple"));
// false       Console.WriteLine(IsPalindrome("Never Odd Or Even")); // true
                Console.WriteLine(IsPalindrome("Curabitur vel est diam")); // false
                Console.WriteLine(IsPalindrome("x"));
// true
        }
    }
}
```

Length of string

```
using System;

namespace CSharpExercises.Strings
{
    class LengthOfAStringTask
    {
        static int LengthOfAString(string str)
        {
            var length = 0;
            foreach (var c in str)
            {
                length++;
            }

            return length;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(LengthOfAString("Lorem ipsum dolor sit amet")); //26
            Console.WriteLine(LengthOfAString(string.Empty));                //0
            Console.WriteLine(LengthOfAString("conse12ctetur "));              //14
        }
    }
}
```

Make uppercase

```
using System;

namespace CSharpExercises.Exercises.Strings
{
    class MakeUppercaseTask
    {
        public static string MakeUppercase(string word)
        {
            var letterIndex = 0;
            var uppercaseWord = string.Empty;

            for (var i = 0; i < word.Length; i++)
            {
                if (word[i] >= 'a' && word[i] <= 'z' && letterIndex % 2 == 0)
                {
                    letterIndex++;
                    uppercaseWord += (char)(word[i] - 32);
                }
                else if (word[i] != ' ')
                {
                    letterIndex++;
                    uppercaseWord += word[i];
                }
                else
                {
                    letterIndex = 0;
                    uppercaseWord += word[i];
                }
            }
            return uppercaseWord;
        }

        public static void Main()
        {
            Console.WriteLine(MakeUppercase("very short sentence.")); // VeRy
            Console.WriteLine(MakeUppercase("motorcycle"));           //
            Console.WriteLine(MakeUppercase("Events And Delegates")); // EvEnTs
        }
    }
}
```

How many occurrences

```
using System;

namespace CSharpExercises.Exercises.Strings
{
    class HowManyOccurrencesTask
    {
        static int HowManyOccurrences(string str, string subStr)
        {
            int found;
            var total = 0;
            for (var i = 0; i < str.Length; i++)
            {
                found = str.IndexOf(subStr, i);

                if (found > -1)
                {
                    total++;
                    i = found;
                }
            }

            return total;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(HowManyOccurrences("He is a good boy, he would never  
do that!", "he"));           // 1
            Console.WriteLine(HowManyOccurrences("104 593 00-930 720193", "93"));
            // 3
            Console.WriteLine(HowManyOccurrences("xyz", "a"));
            // 0
        }
    }
}
```

Sort characters descending

```
using System;

namespace CSharpExercises.Exercises.Strings
{
    class SortCharactersDescendingTask
    {
        static char[] SortCharactersDescending(string str)
        {
            var charArray = str.ToCharArray();
            char ch;

            for (var i = 1; i < str.Length; i++)
            {
                for (var j = 0; j < str.Length - 1; j++)
                {
                    if (charArray[j] < charArray[j + 1])
                    {
                        ch = charArray[j];
                        charArray[j] = charArray[j + 1];
                        charArray[j + 1] = ch;
                    }
                }
            }

            return charArray;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(SortCharactersDescending("Aliquam pulvinar aliquam libero, in fringilla erat.")); // vuutrrrrrqponnnmlllllllliiiiiiiigfeebaaaaaaA.,
            Console.WriteLine(SortCharactersDescending("Thi2 12 5@mpl3 Str!nG~")); // ~trpnmlihTSG@53221!
        }
    }
}
```


Revert words order

```
using System;

namespace CSharpExercises.Exercises.Strings
{
    class RevertWordsOrderTask
    {
        static string RevertWordsOrder(string str)
        {
            var strArray = str.Split(' ');
            var len = strArray.Length;

            for (int i = 0; i < len / 2; i++)
            {
                string temp = strArray[i];

                if (i == 0)
                {
                    strArray[i] = strArray[len - 1].Remove(strArray[len - 1].Length - 1);
                    strArray[len - 1] = temp + strArray[len - 1].Substring(strArray[len - 1].Length - 1);
                }
                else
                {
                    strArray[i] = strArray[len - 1 - i];
                    strArray[len - 1 - i] = temp;
                }
            }

            return string.Join(" ", strArray);
        }

        static void Main(string[] args)
        {
            Console.WriteLine(RevertWordsOrder("Proin in odio viverra, accumsan purus vel, placerat elit!")); // elit placerat vel, purus accumsan viverra, odio in Proin!
            Console.WriteLine(RevertWordsOrder("Cras iaculis tortor justo.")); // justo tortor iaculis Cras.
        }
    }
}
```

Mix two strings

```
using System;

namespace CSharpExercises.Exercises.Strings
{
    class MixTwoStringsTask
    {
        static string MixTwoStrings(string word1, string word2)
        {
            var mixedWord = string.Empty;

            for (var i = 0; i < (word1.Length > word2.Length ? word1.Length :
word2.Length); i++)
            {
                if (i < word1.Length)
                {
                    mixedWord += word1[i];
                }
                if (i < word2.Length)
                {
                    mixedWord += word2[i];
                }
            }

            return mixedWord;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(MixTwoStrings("DoG", "ElEpHaNt"));           //
DEolGEpHaNt
            Console.WriteLine(MixTwoStrings("The tallest", "XXXXXXXXXX")); //
TXhXeX XtXaXlXlXeXsXt
        }
    }
}
```

Number of words

```
using System;

namespace CSharpExercises.Strings
{
    class NumberOfWordsTask
    {
        static int NumberOfWords(string str)
        {
            var numberOfWords = 0;
            for (var i = 1; i < str.Length; i++)
            {
                numberOfWords = (char.IsWhiteSpace(str[i]) ? numberOfWords + 1 :
numberOfWords);
            }

            return numberOfWords + 1;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(NumberOfWords("Mauris consectetur urna sit amet
risus ultricies rutrum.)); // 8
            Console.WriteLine(NumberOfWords("Quisque M"));
// 2
            Console.WriteLine(NumberOfWords("Xor"));
// 1
        }
    }
}
```

String in reverse order

```
using System;

namespace CSharpExercises.Strings
{
    class StringInReverseOrderTask
    {
        static string StringInReverseOrder(string str)
        {
            var reverseString = string.Empty;
            for (var i = str.Length - 1; i >= 0; i--)
            {
                reverseString += str[i];
            }

            return reverseString;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(StringInReverseOrder("Vivamus commodo quam at purus
")); // surup ta mauq odommoc sumaviV
            Console.WriteLine(StringInReverseOrder("34 ( 9 9@*"));
            // *@9 9 ( 43
            Console.WriteLine(StringInReverseOrder("malesuada"));
            // adauselam
        }
    }
}
```

Compress string

```
using System;

namespace CSharpExercises
{
    class Program
    {
        public static string CompressString(string str)
        {
            var count = 0;
            var last = str[0];
            var newStr = string.Empty;

            foreach (var s in str)
            {
                if (s == last)
                {
                    count++;
                }
                else
                {
                    newStr += last.ToString() + count;
                    last = s;
                    count = 1;
                }
            }

            newStr += last.ToString() + count;

            return newStr;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(CompressString("aaaabbccccddaa")); //a4b2c5d1a2
            Console.WriteLine(CompressString("948kro")); //914181k1r1o1
            Console.WriteLine(CompressString("$999j*#jjjfYyyy"));
            // $193j1*1#1j3f1Y1y3
        }
    }
}
```

Sum digits in string

```
namespace CSharpExercises.Exercises.Strings
{
    class SumDigitsInStringTask
    {
        static int SumDigitsInString(string str)
        {
            var sum = 0;

            for (var i = 0; i < str.Length; i++)
            {
                if (char.IsDigit(str[i]))
                {
                    sum += (int)char.GetNumericValue(str[i]);
                }
            }

            return sum;
        }

        public static void Main()
        {
            Console.WriteLine(SumDigitsInString("aaa5aa5aa5a5a")); // 20
            Console.WriteLine(SumDigitsInString("10r3m1p5um")); // 10
            Console.WriteLine(SumDigitsInString("tt")); //0
        }
    }
}
```