Writing to console

The code below uses Console.WriteLine and Console.Write to give the output seen in the picture. Can you figure out what makes these two instruction different?

```
Console.WriteLine("Hello");
Console.WriteLine("world!");

Console.Write("Hello");
Console.Write(" world!");
```

Exercise 1.

Write a simple program that outputs your first and last name to the console on one line each.

Exercise 2.

Write a program that holds 2 string variables. One with your first name and one with your last name. Then output them to the screen.

Exercise 3.

Output todays date to the console. Tip: Use DateTime.Now

What is the benefit of using DateTime.Now instead of just writing the date directly to the console as a string literal?

Exercise 4.

Write a program that holds 2 string variables. One with your first name and one with your last name. Ask the user to input first name and last name and store them in the variables you've created. Then output them to the screen.

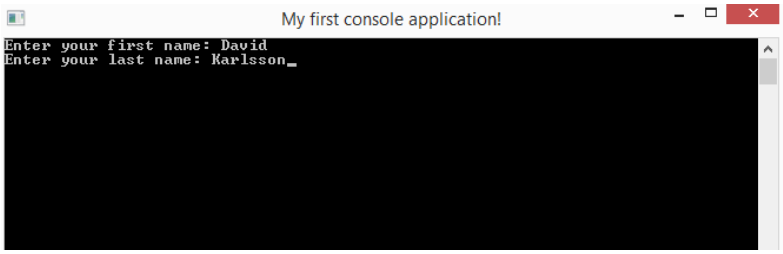
String manipulation

What do you think the following statements will yield for result to the console window?

```
String part = "if you're doing your";
part = part + " best, you won't have any";
String quote = part + " time to worry about failure.";
Console.WriteLine(quote);
```

Exercise 5.

Make a console program that ask the user for his first name, then last name. Store them into one variable each, then concatenate them together and display a message to the user. Use your knowledge with the different Console-methods so that the prompt text to the user and the text he/she inputs are on the same line. Also, make sure your variables have descriptive names.



Exercise 6.

In the previous program you wrote a program that asked the user for his/her first and last name. Alter the program to print the first and last name with upper case characters.

Exercise 7.

Complete the code below so that it prints out "The brown fox jumped over the lazy dog" exactly (all characters must match with lower and upper case). You are not allowed to change the string literal directly in the code below or assign it directly to the string, but must use the string methods to manipulate it to achieve the result. You decide yourself if you want to only manipulate the **str** *variable only or if you want to create new variables to hold temporary values.*

```
String str = "The quick fox Jumped Over the DOG";
 // code here
 Console.ReadLine();
```

Exercise 8.

Below is a given string. Extract the [1,2,3,4,5] *from the string into a new string. Then remove the values 2 and 3 and insert 6-10 into it in the end, comma separated. Display the result.*

```
String str = "Arrays are very common in programming, they look something like: [1,2,3,4,5]";
Console.ReadLine();
```

Exercise 9.

Below is a given string. Extract the four sets (abc), (cba), (bac) and (cab) then print them on one line each in the console window. Tip: We haven't looked at it yet, but you can for example use str.IndexOf("A") + 3 to get back the position 3 step ahead of the first occurrence of A.

```
String str = "(abc) and (cba) are each others reverse! Same goes for (bac) and (cab)!";
Console.ReadLine();
```

Mathematical operators and methods

In the following exercises you are meant to use the common mathematical operators, and the mathematical functions located in the System.Math-class.

Why does following program not output the correct average of the two numbers given?

```
int a = 10;
double b = 2.0;
double average = a + b / 2;
Console.WriteLine(average);
```

Exercise 10.

What is the output of the following code? Try think about it before trying to run the code.

```
int a = 10;
int b = 3;
int c = 7;
int d = 7 / 2.0f;
double result = ((double) a / b) * 2;
Console.WriteLine(result);
result = a / (b + c) + 9.0f;
Console.WriteLine(result);
result = a * a / (b + c);
Console.WriteLine(result);
result = a * (b + a) – 100;
Console.WriteLine(result);
```

Exercise 11.

Let the user input a decimal number. Then output square root of the number and the number raised to the power of 2, 3 and 10. That is $\sqrt{n}, n^2, n^3, n^{10}$.

Selection

Exercise 12.

Write a program that lets a user enter a name. Check if that name is either Bob or Alice. If it is, display "Hi Bob" or "Hi Alice".

Exercise 13.

Write a program that lets a user type in a number and check whether this number is divisible by 4 and display the result to the user.

Exercise 14.

Write a program that lets a user type in some text. Then check whether the word "city" occurs in the string and display a message to the user at what position. Tip: IndexOf

Exercise 15.

Write a program that lets a user type in some text, then check whether the length of the string is greater than 25 characters. Display a message to the user.

Exercise 16.

Write a program that displays 3 different options that the user can choose from. Depending on the choice, display a different message. If the choice is not valid, display a message. Use the Console.Clear-method to remove the options after the choice.

Exercise 17.

*Write a program that lets a user enter two numbers: a and b. Convert them into integers. Then show the result from a + b, a – b, a * b and a / b to the user with a descriptive label. Make a check before dividing that b is not zero, if so display a message that it cannot be divided. Make sure to cast one of the integers to a double number.*

Exercise 18.

Write a program that lets a user type in a number. Check if the number is in the range 10 <= number <= 20. Tip: try using both nested if-statements and && to check the range.

Exercise 19.

Write a program that lets the user input a colour. Then use a switch statement to check if the colour exists. If it doesn't display an error message. Then, print out the name of the colour with the same colours. Tip: use Console.ForegroundColor *to change the color of the text.*

Iteration

Exercise 20.

Write a program that lets the user enter a number. Check that it is greater than zero. Lastly, iterate that many times, and display each number in the iterations. 10 should then display 0,1,2,3,4,5,6,7,8 and 9 in the console window.

Exercise 21.

Write a program that lets the user enter a number. Check that it is greater than zero. Lastly, iterate from that number down to zero and output each number. That means, 10 should give back the result 9,8,7,6,5,4,3,2,1 and 0.

Exercise 22.

Write a program that lets the user enter a number. Check that it is greater than zero. Lastly, iterate that many times, and display each number that is divisible by 2.

Exercise 23.

Write a program that output the multiplication table for 1 to 10.

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

Exercise 24.

Write a program that iterates 1 to 10 and print each number raised to the power 2. Also output text like:

"5 raised to the power of 2 is equal to 25"

Exercise 25.

Write a program that produces the output below using a for-loop

Exercise 26.

Write a program that keeps asking the user for input numbers, until he enters -1.. Store the amount of numbers the user have entered and the sum of the numbers added together. When the user types -1, the program should display the total and the average.

Exercise 27.

Write a program that asks the user for a number. Then display all numbers that the number is divisible by. Example entering 12, should output 6, 4, 3, 2 and 1. Tip: Use modulo.

Exercise 28.

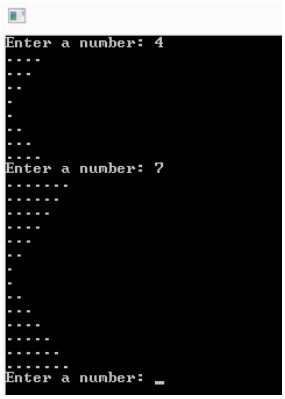
Write a program that outputs the 3 first perfect numbers. A perfect number is a number where all its positive divisors sums up to the actual number. The first number is 28, where 14 + 7 + 4 + 2 + 1 = 28. Tip: look at the previous exercise and build on top of it.

Exercise 29.

Write a program that asks the user for a number. Use this number to output the Fibonacci series up until that number. Entering 10 should then output: 0, 1, 1, 2, 3, 5, 8, 13, 21 and 34.

Exercise 30.

Write a program that asks a user for a number. Use the number to produce following output depending on the number:



Exercise 31.

Let the user input a string, then check if the string is a palindrome. A palindrome is a word or sentence that reads the same in both directions. ***Example Loops at a spool*** *and* ***wet stew****. The spaces between might not be the same thought so keep that in mind.*

Exercise 32.

Implement a console application with a menu system. The menu should contain the four options: 1. Add two numbers together, 2: Repeat a string x times, 3: Convert a string to uppercase and 4: Exit the application. The menu should reside inside a while-loop that repeats each time a menu choice has been completed, hence allowing the user to choose another options. The only exception is the 4th choice that will terminate the loop. Each option should call a method that executes the code for the given option. After each respective method call, the console should clear to avoid cluttering the screen.

Arrays

Exercise 33.

Initiate two arrays, one with integer values, and one with strings. Fill the first one with a couple of numbers, and the second one with names. Tip: use the list initialiser

Exercise 34.

Loop through the arrays from the previous exercise and print the content to the console window. For the names, write every second name with upper case.

Exercise 35.

Ask the user for a number. Use the number to generate an array with the length of the number. Loop through the array and generate a random number on each position. Tip: Use the Random-class.

Exercise 36.

Create an array with arbitrary size and fill with random numbers like the previous exercise. Then create a new array with the same size and copy over the values to it.

Exercise 37.

Create two arrays with arbitrary sizes and fill them with random numbers. Then make a new array with the combined size of the two previous arrays and copy the values into it.

Exercise 38. (Medium)

Create two arrays with arbitrary sizes and fill them with random numbers. Then copy over the numbers so that the even numbers are located in the rear (the right side) part of the array and the odd numbers are located in the front part (the left side).

Exercise 39.

Create two arrays with equal sizes. One should contain username and one should contain passwords. Let the user try to input a username and a password and match it against the arrays. If he types in a correct username and password, display a secret message to him. (You can assume that username on position n belongs to password on position n in respective array)

Exercise 40.

Let the user input a string with numbers comma separated like "1,2,34,83,19,45" then separate the numbers into an array and find the min, max and average value. Print these out to the screen. Tip: use the Split-function.

Exercise 41.

Given strings in the following format "2014-01-22 12:00:00" and "2013-02-10 15:23:00, extract the date part and display it in the following format: "22nd January 2014" and 10th February 2013". Tip: use the Split-function. Note, that this can be done using Date.DateParse, try doing it with string manipulation to practice. When you done, try do it with DateTime.Parse to see the difference.

Exercise 42.

The company See sharp AB have discovered that the users on their forums use a very harsh language when interacting with each other. So now they have asked you to write a swear word filter to censor these occurring words. Before implementing this filter on their website, they want a demonstration in form of a console program.

The program should ask the user for a textual input. Then it should check for all occurrences of swear words. Store the swear words in an array and check the textual input against the array and use string manipulation to replace all swear words with something more appropriate.

Exercise 43.

Given the string "[{ fruit:pear , colour:green }, { fruit:orange, colour: orange },{ fruit:apple, colour : red }]", extract the fruits and their colour and display it to the user like: "The fruit pear is green"

Exercise 44. (Hard)

Create a program that generates 7 unique numbers into an array. The numbers should be between 1 and 40. The numbers may only appear once in the array!

Exercise 45. (Hard)

Create a program that can shuffle a deck of cards. The cards can be represented as an array of integers, like [1,1,1,1,2,2,2,2,3,3,…n]. Then make it possible to draw 1 card from the deck and add to another array. (The card should then disappear from the card deck and appear in the array with the drawn cards. Tip: You can use Array.Resize(ref array, newSize) to change the size of an existing array, and Array.Copy. There is however not a requirement that the array needs to be in a specific order after a card has been drawn. Complete the functions below.

```
static int DrawCard(ref int[] deck)
{
   // code to draw a card here
}

static void ShuffleCards(int[] deck)
{
   // code to shuffle the deck here
}
```