# Homework: Console Input / Output

This document defines homework assignments from the [“C# Basics“ Course @ Software University](http://softuni.bg/courses/csharp-basics/). Please submit as homework a single zip / rar / 7z archive holding the solutions (source code only) of all below described problems.

## Sum of 3 Numbers

Write a program that reads **3 real numbers** from the console and prints their **sum**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **c** | **sum** |
| 3 | 4 | 11 | 18 |
| -2 | 0 | 3 | 1 |
| 5.5 | 4.5 | 20.1 | 30.1 |

## Print Company Information

A company has **name, address, phone number, fax number, web site and manager**. The manager has **first name, last name, age and a phone number**. Write a program that reads the information about a company and its manager and prints it back on the console.

|  |  |
| --- | --- |
| **program** | **user** |
| Company name: | Software University |
| Company address: | 26 V. Kanchev, Sofia |
| Phone number: | +359 899 55 55 92 |
| Fax number: |  |
| Web site: | <http://softuni.bg> |
| Manager first name: | Svetlin |
| Manager last name: | Nakov |
| Manager age: | 25 |
| Manager phone: | +359 2 981 981 |
| Software University  Address: 26 V. Kanchev, Sofia  Tel. +359 899 55 55 92  Fax: (no fax)  Web site: <http://softuni.bg>  Manager: Svetlin Nakov (age: 25, tel. +359 2 981 981) |  |

## Circle Perimeter and Area

Write a program that reads the radius **r** of a circle and prints its perimeter and area formatted with 2 digits after the decimal point. Examples:

|  |  |  |
| --- | --- | --- |
| **r** | **perimeter** | **area** |
| 2 | 12.57 | 12.57 |
| 3.5 | 21.99 | 38.48 |

## Number Comparer

Write a program that gets **two numbers** from the console and prints the greater of them. Try to implement this without **if** statements. Examples:

|  |  |  |
| --- | --- | --- |
| **a** | **b** | **greater** |
| 5 | 6 | 6 |
| 10 | 5 | 10 |
| 0 | 0 | 0 |
| -5 | -2 | -2 |
| 1.5 | 1.6 | 1.6 |

## Formatting Numbers

Write a program that reads 3 numbers: an integer a (0 ≤ a ≤ 500), a floating-point b and a floating-point c and **prints them in 4 virtual columns** on the console. Each column should have a width of 10 characters. The number a should be printed in **hexadecimal, left aligned**; then the number a should be printed in binary form, padded with zeroes, then the number b should be **printed with 2 digits after the decimal point, right aligned**; the number c should be **printed with 3 digits after the decimal point, left aligned**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **c** | **result** |
| 254 | 11.6 | 0.5 | |FE |0011111110| 11.60|0.500 | |
| 499 | -0.5559 | 10000 | |1F3 |0111110011| -0.56|10000 | |
| 0 | 3 | -0.1234 | |0 |0000000000| 3|-0.123 | |

## Quadratic Equation

Write a program that reads the coefficients **a**, **b** and **c** of a quadratic equation **ax2 + bx + c = 0** and solves it (prints its real roots). Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **c** | **roots** |
| 2 | 5 | -3 | x1=-3; x2=0.5 |
| -1 | 3 | 0 | x1=3; x2=0 |
| -0.5 | 4 | -8 | x1=x2=4 |
| 5 | 2 | 8 | no real roots |

## Sum of 5 Numbers

Write a program that **enters 5 numbers** (given in a single line, separated by a space), **calculates and prints their sum**. Examples:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **numbers** | **sum** |  | **numbers** | **sum** |  | **numbers** | **sum** |
| 1 2 3 4 5 | 15 | 10 10 10 10 10 | 50 | 1.5 3.14 8.2 -1 0 | 11.84 |

## Numbers from 1 to n

Write a program that reads an integer number **n** from the console and prints all the numbers in the interval [**1**..**n**], each on a single line. Note that you may need to use a for-loop. Examples:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **numbers** | **sum** |  | **numbers** | **sum** |  | **numbers** | **sum** |
| 3 | 1  2  3 | 5 | 1  2  3  4  5 | 1 | 1 |

## Sum of n Numbers

Write a program that enters a number **n** and after that enters more **n** numbers and calculates and prints their sum. Note that you may need to use a for-loop. Examples:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **numbers** | **sum** |  | **numbers** | **sum** |  | **numbers** | **sum** |
| 3  20  60  10 | 90 | 5  2  -1  -0.5  4  2 | 6.5 | 1  1 | 1 |

## Fibonacci Numbers

Write a program that reads a number n and prints on the console the first n members of the [**Fibonacci sequence**](http://en.wikipedia.org/wiki/Fibonacci_number) (at a single line, separated by spaces) : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, …. Note that you may need to learn how to use loops. Examples:

|  |  |
| --- | --- |
| **n** | **comments** |
| 1 | 0 |
| 3 | 0 1 1 |
| 10 | 0 1 1 2 3 5 8 13 21 34 |

## \* Numbers in Interval Dividable by Given Number

Write a program that reads two positive integer numbers and prints how many numbers **p** exist between them such that the reminder of the division by **5** is **0**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **start** | **end** | **p** | **comments** |
| 17 | 25 | 2 | 20, 25 |
| 5 | 30 | 6 | 5, 10, 15, 20, 25, 30 |
| 3 | 33 | 6 | 5, 10, 15, 20, 25, 30 |
| 3 | 4 | 0 | - |
| 99 | 120 | 5 | 100, 105, 110, 115, 120 |
| 107 | 196 | 18 | 110, 115, 120, 125, 130, 135, 140, 145, 150, 155, 160, 165, 170, 175, 180, 185, 190, 195 |

## \*\* Falling Rocks

Implement the "**Falling Rocks**" game in the text console. A small dwarf stays at the bottom of the screen and can move left and right (by the arrows keys). A number of rocks of different sizes and forms constantly fall down and you need to avoid a crash.

Rocks are the symbols **^**, **@**, **\***, **&**, **+**, **%**, **$**, **#**, **!**, **.**, **;**, **-** distributed with appropriate density. The dwarf is **(O)**. Ensure a constant game speed by **Thread.Sleep(150)**.

Implement collision detection and scoring system.



# Exam problems.\*\*

All of the problems below are given from the previous C# Basics exams. **You are not obligated** to submit any of them in your homework. We highly recommend you to try solving some or all of them so you can be well prepared for the upcoming exam. You need to learn how to use conditional statements, loops, arrays and other things (learn in internet how or read those chapters in the book “[Fundamentals of computer programming with C#](http://www.introprogramming.info/intro-csharp-book/read-online/)”). If you still find those problems too hard for solving it’s very useful to **check** and **understand** the solutions. You can download all solutions and tests for this variant [here](https://softuni.bg/downloads/svn/csharp-basics/Feb-2014/9.%20CSharp-Basics-Exam-April-2014-Variant-1.zip) or check all [previous exams](https://softuni.bg/trainings/coursesinstances/details/2) (scroll down to the bottom of the page). You can also test your solutions in our automated [judge system](http://judge.softuni.bg/Contests/2/CSharp-Basics-Exam-10-April-2014-Morning) to see if you pass all tests.

## \* – Work Hours

**This problem is from Variant 3 of C# Basics exam from 11-04-2014 Morning. You can test your solution** [**here**](http://judge.softuni.bg/Contests/4/CSharp-Basics-Exam-11-April-2014-Morning) **.**

Lelia Vanche is a very keen freelance developer. She is well known for her outstanding skills, but she is also known for being pretty moody, which often affects her productivity. She also has a passion for bicycles and **10% of the normal work days** she goes mountain-biking instead of working.

You are asked to calculate whether Lelia Vanche can finish a project on time. You will be given the number of **hours required to finish the project**, the **days** that Lelia Vanche has available for working (mind that she goes to biking in 10% of this time) and her **average productivity** during the given period. Assume that a normal work day for Lelia Vanche has **12 hours**. Note that only the whole hours are taken (e.g. 6.98 hours is rounded down to 6 hours).

### Input

Input data should be read from the console.

* The number **h** (the required work **hours** to finish the project) is on the first input line.
* The number **d** (the **days** available to finish the project) is on the second input line.
* The number **p** (the productivity in **percent**) is on the third input line.

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

The output data must be printed on the console.

* On the first output line you should print ‘**Yes’** or ‘**No’** if Lelya Vanche can complete the project.
* On the second output line you should print the **difference,** between the project hours and the work hours**.**

### Constraints

* The number **h** will be an integer between 0 and 2 147 483 647, inclusive.
* The number **d** will be an integer between 0 and 89 478 485, inclusive.
* The number **p** will be an integer between 0 and 100, inclusive.
* Allowed working time for your program: 0.1 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 60  6  75 | No  -12 | The project requires 60 hours. Lalia Vanche has 6 days, of which 10% she will be biking, so she will work 5.4 days \* 12 hours = 64.8 hours \* 75% productivity = 48.6 efficient work hours, which is rounded down to 48. She will be unable to complete the project. The difference is 60 - 48 = -12 (she needs more 12 hours). |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 1  1  100 | Yes  9 | 240  10  100 | No  -132 | 10  10  10 | Yes  0 | 21  10  10 | No  -11 |

## \*\*– Sum of Elements

**This problem is from Variant 3 of C# Basics exam from 11-04-2014 Morning. You can test your solution** [**here**](http://judge.softuni.bg/Contests/4/CSharp-Basics-Exam-11-April-2014-Morning) **.**

You are given **n** numbers. Find an **element that is equal to the sum of all of the other elements**.

### Input

Input data should be read from the console.

* At the **only line** in the input a **sequence of integers** stays (numbers separated one from another by a space).

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

The output data must be printed on the console. At the only line of the output print the result.

* Print "**Yes, sum=…**" if there is an element that is equal to the sum of all other elements, along with this **sum**.
* Print "**No, diff=…**" if there is no element that is equal to the sum of all other elements. Print also the minimum possible difference between an element from the sequence and the sum of all other elements (always a **positive number**).

### Constraints

* All input numbers are integers in the range [0 … 2 000 000 000].
* The count **n** of the input integers is in the range [2 ... 1 000].
* Allowed working time for your program: 0.1 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Output** | **Comments** |  | **Input** | **Output** |
| 3 4 1 1 2 12 1 | Yes, sum=12 | 3 + 4 + 1 + 2 + 1 + 1 = 12 | 6 1 2 3 | Yes, sum=6 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 1 1 10 | No, diff=8 | 5 5 1 | No, diff=1 | 1 1 1 | No, diff=1 | 0 0 | Yes, sum=0 |

## \* – New House

**This problem is from Variant 4 of C# Basics exam from 11-04-2014 Evening. You can test your solution** [**here**](http://judge.softuni.bg/Contests/5/CSharp-Basics-Exam-11-April-2014-Evening) **.**

Little Joro likes to build huts. After he built all the huts in his whole village, he decided to go to the big city and start building houses. But his knowledge of how to do this is limited. Help Joro to design the façade of a beautiful house by printing it to the console. The house must have a roof and one floor. The roof must contains only the symbols ‘**\***’ and ‘**-**’ and the floor must contains the symbols ‘**\***’ and ‘**|**’ (see the examples below).

### Input

* The input data should be read from the console.
* At the only input line you are given an integer number **N** (always an **odd number**) showing the height of the house (without the roof).

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

* The output should be printed on the console.
* You should print the house on the console, just like in the examples below. Each row can contain only the following characters: “**-**” (dash), “**\***” (asterisk) and “**|**” (pipe).

### Constraints

* The number **N** will be a positive odd integer number between 3 and 101, inclusive.
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 3 | -\*-  \*\*\*  |\*|  |\*|  |\*| | 5 | --\*--  -\*\*\*-  \*\*\*\*\*  |\*\*\*|  |\*\*\*|  |\*\*\*|  |\*\*\*|  |\*\*\*| | 7 | ---\*---  --\*\*\*--  -\*\*\*\*\*-  \*\*\*\*\*\*\*  |\*\*\*\*\*|  |\*\*\*\*\*|  |\*\*\*\*\*|  |\*\*\*\*\*|  |\*\*\*\*\*|  |\*\*\*\*\*|  |\*\*\*\*\*| |

## \*\* – Magic Strings

**This problem is from Variant 3 of C# Basics exam from 11-04-2014 Morning. You can test your solution** [**here**](http://judge.softuni.bg/Contests/4/CSharp-Basics-Exam-11-April-2014-Morning) .

You are given a number **diff**. Write a program to **generate all sequences of 8 letters**, each from the set { '**s**', '**n**', '**k**', '**p**' }, such that the weight of the first 4 letters differs from the weight of the second 4 letters exactly by **diff**. These sequences are called “**magic strings**”. Print them in alphabetical order.

The **weight of a single letter** is calculated as follows: weight('**s**') = **3**; weight('**n**') = **4**; weight('**k**') = **1**; weight('**p**') = **5**. The **weight of a sequence** of 4 letters is the calculated as sum of the weights of these 4 individual letters.

### Input

* The input data should be read from the console.
* The number **diff** stays at the first line.

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

The output should be printed on the console as a sequence of strings in **alphabetical order**. Each string should stay on a separate line. In case no magic strings exist, print “**No**”.

### Constraints

* The number **diff** will be an **integer** number in the range [0…100].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 16 | kkkkpppp  ppppkkkk | weight('kkkk') = 4; weight('pppp') = 20; diff = 16  weight('pppp') = 20; weight('kkkk') = 4; diff = 16 |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 15 | kkkknppp  kkkkpnpp  kkkkppnp  kkkkpppn  npppkkkk  pnppkkkk  ppnpkkkk  pppnkkkk | weight('kkkk') = 4; weight('nppp') = 19; diff = 15  weight('kkkk') = 4; weight('pnpp') = 19; diff = 15  weight('kkkk') = 4; weight('ppnp') = 19; diff = 15  weight('kkkk') = 4; weight('pppn') = 19; diff = 15  weight('nppp') = 19; weight('kkkk') = 4; diff = 15  weight('pnpp') = 19; weight('kkkk') = 4; diff = 15  weight('ppnp') = 19; weight('kkkk') = 4; diff = 15  weight('pppn') = 19; weight('kkkk') = 4; diff = 15 |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 20 | No | No magic strings match the specified difference diff |

## – Catch the Bits

**This problem is from Variant 4 of C# Basics exam from 11-04-2014 Evening. You can test your solution** [**here**](http://judge.softuni.bg/Contests/5/CSharp-Basics-Exam-11-April-2014-Evening) .

You are given a **sequence of bytes**. Consider each byte as sequence of exactly 8 bits. You are given also a number **step**. Write a program to extract all the bits at positions: **1**, **1 + step**, **1 + 2\*step**, ... Print the output as a sequence of bytes. In case the last byte have less than 8 bits, add trailing zeroes at its right end. Bits in each byte are counted from the leftmost to the rightmost. Bits are numbered starting from 0.

### Input

* The input data should be read from the console.
* The number **n** stays at the first line.
* The number **step** stays at the second line.
* At each of the next **n** lines **n** bytes are given, each at a separate line.

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

The output should be printed on the console. Print the output bytes, each at a separate line.

### Constraints

* The number **n** will be an **integer** number in the range [1…100].
* The number **step** will be an **integer** number in the range [1…20].
* The **n numbers** will be integers in the range [0…255].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 2  11  109  87 | 128 | We have the following input sequence of 16 bits (2 bytes):  0**1**101101 0101**0**111.  We take the bits 1 and 12 (step=11). We obtain the sequence **10**.  We pad the sequence with 6 trailing zeroes. Result: **10000000**. |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 3  2  45  87  250 | 63  192 | We have the following input sequence of 24 bits (3 bytes):  0**0**1**0**1**1**0**1** 0**1**0**1**0**1**1**1** 1**1**1**1**1**0**1**0**  We take bits 1, 3, 5, …, 23 (step=2). We obtain the sequence:  00111111 1100. We pad it with 4 zeroes to obtain 2 full bytes. Result: **00111111 11000000**. |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 2  2  45  87 | 63 | We have the following input sequence of 16 bits (2 bytes):  0**0**1**0**1**1**0**1** 0**1**0**1**0**1**1**1**  We take bits 1, 3, 5, …, 15 (step=2). We obtain the sequence:  00111111 (8 bits). No padding is needed. Result: **00111111**. |