# 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: | 15-18 Tintyava, 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| 11.6|0.500 | |
| 499 | -0.5559 | 10000 | |1F3 |0111110011| -0.56|10000.000 | |
| 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 exam (29-March-2015-Evening). **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://judge.softuni.bg/Contests/79/Programming-Basics-Exam-29-March-2015-Evening) or check all [previous exams](https://softuni.bg/courses/programming-basics). You can also test your solutions in our automated [judge system](https://judge.softuni.bg/Contests/79/Programming-Basics-Exam-29-March-2015-Evening) to see if you pass all tests.

## \*\* Currency Check

Te4o is a big Battlefield fan. He's been saving money for months to buy the new Battlefield Hardline game. However, he has **five options** to buy the game from. **The first one** is a shady Russian site selling games in **rubles** (Russian currency). **Another option** is an American site selling games in **dollars** (American currency). Te4o's **third option** is the official site of the game - selling games in **euros** (European Union currency). **The final 2 options** are Bulgarian sites **B** and **M.** Both of them sell in **leva** (Bulgarian currency). **B** offersa very **special deal - 2 copies** of the gamefor **the price of one. M** sellsgamesfor **normal prices.** Te4o is very bad with math and can't calculate the game prices in leva. But he wants to impress his girlfriend by showing her he bought the cheapest game.

Assume that Te4o has a girlfriend, **all games are identical, 100 rubles** are **3.5 leva, 1 dollar** is **1.5 leva, 1 euro** is **1.95 leva** and if Te4o **buys 2 special games** fromB he can **sell one** of them for exactly **half of the money** he paid for both**.** Your task is to write a program that calculates **the cheapest game**.

### Input

The input data should be read from the console. It consists of five input values, each at a separate line:

* The number **r** – amount of **rubles** Te4o has to pay for the game at the Russian site.
* The number **d** – amount of **dollars** Te4o has to pay for the game at the American site.
* The number **e** – amount of **euro** Te4o has to pay for the game at the official site.
* The number **b** – amount of **leva** Te4o has to pay for the special offer at B.
* The number **m** – amount of **leva** Te4o has to pay for the game at M's site.

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 only output line you must print **the cheapest game price rounded up (removed "up") to the second digit after the decimal mark.**

### Constraints

* The numbers **r, d, e, b, m** are integer numbers in range [0... 4,294,967,295].
* Allowed working time for your program: 0.1 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** | **Comments** | | | | | |
| 2600  60  60  130  70 | 65.00 | 2600 rubles is 2600 / 100 = 26 \* 3.5 = 91 leva, 60 dollars is 60 \* 1.5 = 90 leva , 60 euro is 60 \* 1.95 = 117 leva, 130 leva for 2 games is 130 / 2 = 65 leva per game and 70 leva is 70 leva. The cheapest game is 65 leva. | | | | | |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 2050  85  75  239  80 | 71.75 | 800  11  15  28  13 | 13.00 | 700  5  1  20  4 | 1.95 |

## \*\* Dream Item

Trifon has a dream item. He is a good programmer and started working last month, but he needs help with his salary**.** You are given **the month** when Trifon is working, themoneyhe is **making** **per hour, number of hours per day** he is workingand the **price** of his dream item. Assume **February** has **28 days** and every month has **exactly 10 holidays** when Trifon is not working. All other months have either 31 or 30 (check a calendar if you’re unsure about the number of days in a given month). Also ifTrifon makes **more** than **700** leva this month, he is promised a **bonus of 10%** of the total money (e.g. if he makes 800 lv, his bonus will be 80 and the total money he would earn is 880 lv).

Your task is to write a program that calculates whetherTrifon **can buy his dream item**.

### Input

The input data consists of one line coming from the console. Check the examples below.

The format is: ***Month\Money per hour\Hours per day\Price of the item***.

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 only output line you must print whether the item can be bought - **"Money left = {0} leva."** or **"Not enough money. {0} leva needed."**
* The money must be **rounded to the second digit** after the decimal point.

### Constraints

* Month - "Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug", "Sept", "Oct", "Nov" or "Dec".
* Money per hour - floating number [-7.9 x 1028 … 7.9 x 1028].
* Hours per day - integer in range [1 to 24].
* Item price - floating number in range [0... +7.9 x 1028].
* Allowed memory: 16 MB. Allowed time: 0.1 seconds.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| Apr\2.6\7\300 | Money left = 64.00 leva. | 20 working days \* 2.6 lv per hour \* 7 hours = 364 lv - 300 lv item price = 64 lv. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** |
| Jan\9.234\6\2000 | Not enough money. 720.17 leva needed. | Apr\5.6\10\1000 | Money left = 232.00 leva. |

## \*\* Magic Wand

As we all know programmers often make mistakes in their code. They spend hours and hours trying to figure out where the problem is. Some are praying for the code to fix itself, others are searching for magical rainbow unicorns to help them with their problem. One day, the programmers Gesho and Posho discovered a way to build magic wands that solve their coding problems. Your task is to help Gesho and Posho to build a **magic wand**.

### Input

The input data should be read from the console.

On the only input line you have an integer number **N**. The **width** of the wand is **3\*N+2**.

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

### Output

The output data should be printed on the console. You must print the magic wand on the console.

Each row contains characters "**.**" (dot) and "**\***" (asterisk).

### Constraints

* The number **N** will always be an **odd** integer number in the range [5…39].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

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

## \*\* Array Matcher

You are given a **string** that consists of **two character arrays** and **a command**. Your task is to create a **new array** from the given two by executing the command.

If the command says **"join"** it means that you should create an array with elements that are contained in both arrays. If the command says "**right exclude**" it means that the newly created array should contain only elements from the first array that are not contained in the second array. If the command says "**left exclude**" it means that you should create an array with elements from the second array that are not contained in the first array.

The newly created array should have its elements sorted by their ASCII code. Examples:

1. You are given the array "ABCD", the array "CAFG" and the command "join". The new array should be "AC".
2. You are given the array "ABCD", the array "CAFG" and the command "right exclude". The new array should be "BD".
3. You are given the array "ABCD", the array "CAFG" and the command "left exclude". The new array should be "FG".

### Input

The input data should be read from the console.

* A **single line** containing the **two arrays** and the **command**, separated by a '\' sign.

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. It should consist of **exactly 1 line**:

* The output should be the elements of the newly formed array, sorted by their ASCII code.

### Constraints

* The characters of the arrays will be **characters** from the **ASCII** table.
* Each element in an array will have only **one occurrence**.
* Allowed working time for your program: 0.1 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ABCD\CAFG\join | AC |

|  |  |
| --- | --- |
| **Input** | **Output** |
| EDCBA\ZHGLCA\left exclude | GHLZ |

## \*\* Knight Path

You are given a chess board of **size** **8 x 8**, divided into 64 cells. Initially, the board is empty (all cells have a value of **0**). You place a knight on the board which inverts the bits on all positions it lands on (0 -> 1; 1 -> 0). The initial position of the knight is [0, 0]. Each move will be defined by a combination of two directions, one vertical and one horizontal (example: "left up"). The knight moves according to the standard rules of chess – two positions in the first direction and then one position in the second direction (see the example below for reference). When the given command leads you out of the board don’t move the knight. The valid commands are:

"**left up**", "**left down**", "**right up**", "**right down**", "**up left**", "**up right**", "**down left**", "**down right**".

When you receive the string "**stop**" from the console, you should stop moving the knight. There would be some 1s and 0s on the board. Each row of the board represents a binary integer number. Your task is to print all the rows, which are different from 0, to the console. To understand the task better check the example below.

### Input

The input data is read from the console.

* It consists of a **random number of lines**. The input **ends with the string "stop"**.
* Each line will hold **a string -** representing the **direction** of the **knight’s** movement; the vertical and horizontal directions will be separated from each other by a single 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.

* On the only output line you must print the non-zero integers from the board**.**
* Each row of the table represents an integer number in binary format.
* If the **whole board** is with zeroes, you should print out "[Board is empty]".

### Constraints

* The **number moves** will be in the range [1…25].
* The **direction** will consist of a combination of the following strings: **"left", "right", "up", "down"**.
* Time limit: 0.1 seconds. Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Number |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 |
| 7 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 16 |

|  |  |
| --- | --- |
| **Input** | **Output** |
| left down | 1 |
| down right | 4 |
| right up | 2 |
| down right | 1 |
| left down | 4 |
| left down | 16 |
| stop |  |