# Homework: Introduction to Mathematics

This document defines homework assignments from the [“C# Basics“ Course @ Software University](http://softuni.bg/courses/csharp-basics/). Please submit as homework a single txt/doc/docx file holding the answers of all below described problems.

## Some Primes

Find the 24th, 101st and 251st prime number.

## Some Fibonacci Primes

Check if the 24th, 101st and 251st prime numbers are part of the base Fibonacci number set. What is their position?

## Some Factorials

Find 100!, 171! and 250! Give all digits.

## Calculate Hypotenuse

You are given three right angled triangles. Find the length of their hypotenuses.

1. Catheti: 3 and 4
2. Catheti: 10 and 12
3. Catheti 100 and 250

## Numeral System Conversions

Convert 1234d to binary and hexadecimal numeral systems.

Convert 1100101b to decimal and hexadecimal numeral systems.

Convert ABChex to decimal and binary numeral systems.

## Least Common Multiple

Find LCM(1234, 3456).

# Homework: Introduction to Programming

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.

## Play with Visual Studio

Familiarize yourself with Microsoft Visual Studio (if you already have it installed) or **install Visual Studio** (or Visual Studio Express) at your laptop or home computer. Search in Internet for the correct download link: <http://google.com/search?q=download+visual+studio>. You do not have to submit anything in your homework for this problem.

Start Visual Studio and play with it. **Create a simple C# program** (console application), compile and run it.

## Blank Solution in Visual Studio

Create a **blank solution** in Visual Studio called “Intro-Programming-Homework”. This solution will hold all your homework projects, code and files. For each problem (exercises) add a separate project with self-descriptive name like “Hello-World” and “Print-Your-Name”. You do not have to submit anything in your homework for this problem.

## Play with MSDN Library

Play with Microsoft Developer Network (MSDN) Library Documentation. You may find it online at <http://msdn.microsoft.com/library>.

* Find information about Console.WriteLine() method in MSDN.
* Find information about the Console class.
* Find information about the class keyword.

You do not have to submit anything in your homework for this problem.

## Hello World

Create, compile and run a **“Hello C#” console application**. Ensure you have named the application well (e.g. “”HelloCSharp”). You should submit the Visual Studio project holding the HelloCSharp class as part of your homework.

## Print Your Name

Modify the previous application to **print your name**. Ensure you have named the application well (e.g. “PrintMyName”). You should submit a separate project Visual Studio project holding the PrintMyName class as part of your homework.

## Print Numbers

Write a program to print the numbers 1, 101 and 1001, each at a separate line. Name the program correctly. You should submit in your homework the Visual Studio project holding the source code of the PrintNumbers program.

## Print First and Last Name

Create console application that **prints your first and last name**, each at a separate line.

## Square Root

Create a console application that calculates and prints the **square root** of the number 12345. Find in Internet “how to calculate square root in C#”.

## Print a Sequence

Write a program that prints the first 10 members of the sequence: 2, -3, 4, -5, 6, -7, ...

## Reformat C# Code

Reformat the following C# code to **make it readable** according to the C# best practices for code formatting. Change the casing of the identifiers in the code (e.g. use PascalCase for the class name):

|  |
| --- |
| HorribleCode.cs |
| using  System;  class hoRRiblEcoDe  {  static  void  Main()  {  Console.  WriteLine("Hi, I am horribly formatted program"  ); Console.  WriteLine("Numbers and squares:")  ; for (int i = 0;  i < 10;  i++)  {  Console.WriteLine(i +  " --> " + i  \*  i);  }  }  } |

## Programming Languages

Perform a research (e.g. in Google or Wikipedia) and provide a short list with information about the most popular programming languages. How similar are they to C#? How do they differ from C#? Write in a text file called “programming-languages.txt” at least five languages along with 2-3 sentences about each of them. Use English.

## Development Environments

Perform a research (e.g. in Google or Wikipedia) and provide a short list with popular development environments (IDEs) like Visual Studio. Write in a text file called “list-of-IDEs.txt” at least five IDEs along with 2-3 sentences about each of them. Use English.

## C# and .NET Differences

Describe the difference between C# and .NET Framework in 2-3 sentences. Write your description in a text file called “csharp-and-dot-net-framework.txt”. Use English.

## \* Current Date and Time

Create a console application that **prints the current date and time**. Find in Internet how.

## \* Age after 10 Years

Write a program to read your birthday from the console and print how old you are now and how old you will be after 10 years.

## \* Print Long Sequence

Write a program that prints the first 1000 members of the sequence: 2, -3, 4, -5, 6, -7, … You might need to learn how to use loops in C# (search in Internet).

## \* Play with the Debugger in Visual Studio

Write a program that prints at the console the numbers from 1 to 1000, each at a separate line. You might need to learn how to use loops (search in Internet). Set a **breakpoint** in the line, which prints the next number in the Visual Studio code editor. Run the program through the debugger using the [F5] key. When the debugger stops at the breakpoint trace the code execution with [F10] key. You do not have to submit anything in your homework for this problem. Just play with the debugger to learn how it works.

# Homework: Primitive Data Types and Variables

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.

## Declare Variables

Declare five variables choosing for each of them the most appropriate of the types byte, sbyte, short, ushort, int, uint, long, ulong to represent the following values: 52130, -115, 4825932, 97, -10000. Choose a large enough type for each number to ensure it will fit in it. Try to compile the code. Submit the source code of your Visual Studio project as part of your homework submission.

## Float or Double?

Which of the following values can be assigned to a variable of type float and which to a variable of type double: 34.567839023, 12.345, 8923.1234857, 3456.091? Write a program to assign the numbers in variables and **print** them to ensure no precision is lost.

## Variable in Hexadecimal Format

Declare an integer variable and assign it with the value 254 in hexadecimal format (0x##). Use Windows Calculator to find its hexadecimal representation. Print the variable and ensure that the result is “254”.

## Unicode Character

Declare a character variable and assign it with the symbol that has Unicode code 42 (decimal) using the '\u00XX' syntax, and then print it. Hint: first, use the Windows Calculator to find the hexadecimal representation of 42. The output should be “\*”.

## Boolean Variable

Declare a Boolean variable called isFemale and assign an appropriate value corresponding to **your gender**. Print it on the console.

## Strings and Objects

Declare two **string variables** and assign them with “Hello” and “World”. Declare an **object variable** and assign it with the **concatenation** of the first two variables (mind adding an interval between). Declare a third string variable and initialize it with the value of the object variable (you should perform type **casting**).

## Quotes in Strings

Declare two string variables and assign them with following value:

|  |
| --- |
| The "use" of quotations causes difficulties. |

Do the above in two different ways: with and without using **quoted strings**. Print the variables to ensure that their value was correctly defined.

## Isosceles Triangle

Write a program that prints an isosceles triangle of 9 copyright symbols ©, something like this:

|  |
| --- |
| ©  © ©  © ©  © © © © |

Note that the © symbol may be displayed incorrectly at the console so you may need to change the console character encoding to UTF-8 and assign a Unicode-friendly font in the console. Note also, that under old versions of Windows the © symbol may still be displayed incorrectly, regardless of how much effort you put to fix it.

## Exchange Variable Values

Declare two integer variables a and b and assign them with 5 and 10 and after that exchange their values by using some programming logic. Print the variable values before and after the exchange.

## Employee Data

A marketing company wants to keep record of its employees. Each record would have the following characteristics:

* First name
* Last name
* Age (0...100)
* Gender (m or f)
* Personal ID number (e.g. 8306112507)
* Unique employee number (27560000…27569999)

Declare the variables needed to keep the information for a single employee using appropriate primitive data types. Use descriptive names. **Print** the data at the console.

## Bank Account Data

A bank account has a holder name (first name, middle name and last name), available amount of money (balance), bank name, IBAN, 3 credit card numbers associated with the account. Declare the variables needed to keep the information for a single bank account using the appropriate data types and descriptive names.

## Null Values Arithmetic

Create a program that assigns null values to an integer and to a double variable. Try to print these variables at the console. Try to add some number or the null literal to these variables and print the result.

## \* Comparing Floats

Write a program that **safely compares floating-point numbers** (double) with precision eps = 0.000001. Note that we cannot directly compare two floating-point numbers a and b by a==b because of the nature of the floating-point arithmetic. Therefore, we assume two numbers are equal if they are more closely to each other than a fixed constant eps. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **Number a** | **Number b** | **Equal (with precision eps=0.000001)** | **Explanation** |
| 5.3 | 6.01 | false | The difference of 0.71 is too big (> eps) |
| 5.00000001 | 5.00000003 | true | The difference 0.00000002 < eps |
| 5.00000005 | 5.00000001 | true | The difference 0.00000004 < eps |
| -0.0000007 | 0.00000007 | true | The difference 0.00000077 < eps |
| -4.999999 | -4.999998 | false | Border case. The difference 0.000001 == eps. We consider the numbers are different. |
| 4.999999 | 4.999998 | false | Border case. The difference 0.000001 == eps. We consider the numbers are different. |

## \* Print the ASCII Table

Find online more information about [**ASCII**](http://www.ascii-code.com/) (American Standard Code for Information Interchange) and write a program to prints the entire ASCII table of characters at the console (characters from 0 to 255). Note that some characters have a special purpose and will not be displayed as expected. You may skip them or display them differently. You may need to use for-loops (learn in Internet how).

# 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, but it is highly recommend that you solve some or all of them so you can be well prepared for the upcoming exam. You may 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.

## \* Joro, the Football Player

This problem comes from C# Basics practical exam (10 April 2014 Morning). You may submit your solution here: <http://judge.softuni.bg/Contests/2/CSharp-Basics-Exam-10-April-2014-Morning>.

Joro loves a lot to play football. He used to play for his local village club “Pantera” Kaloyanovets. However, he is a programmer now and he is very busy. Now he is able to **play only in the holidays** and in the **weekends**. Joro plays in **1/2 of the holidays** and **twice in the weekends**: each **Saturday** and each **Sunday**, but **not every weekend** – only when he is **not tired** and only when he is **not going to his hometown**. Joro goes at his hometown **h** weekends in the year. The other weekends are considered “**normal**”. Joro is **tired in 1/3 of the normal weekends**. When Joro is at his hometown, he always plays football with his old friends once, at **Sunday**. In addition, if the year is **leap**, Joro plays football **3 more times** additionally, in non-weekend days. We assume the year has **exactly 52 weekends**.

Your task is to write a program that calculates **how many times Joro plays football** (rounded down to the nearest integer number).

### Input

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

* The string “**t**” for leap year or “**f**” for year that is not leap.
* The number **p** – number of **holidays** in the year (which are not Saturday or Sunday).
* The number **h** – number of weekends that Joro spends in his **hometown**.

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 an integer representing how many times Joro plays football for a year.

### Constraints

* The numbers **p** is in range [0...300] and **h** is in range [0…52].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| t  1  2 | 38 | 52 weekends total in the year, split into:   * 2 hometown weekends 🡪 2 Sundays 🡪 **2** plays * 50 normal weekends 🡪 50 \* 2 / 3 🡪 **33.33** plays   1 holiday 🡪 **0.5** plays  Leap years 🡪 additional **3** plays  Total plays = **38.83** plays 🡪 **38** (rounded) |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| f  3  2 | 36 | t  2  3 | 39 | f  10  5 | 41 | t  0  1 | 38 |

## \* Half Sum

This problem comes from C# Basics practical exam (10 April 2014 Morning). You may submit your solution here: <http://judge.softuni.bg/Contests/2/CSharp-Basics-Exam-10-April-2014-Morning>.

Nakov likes numbers. He often plays with their sums and differences. Once he invented the following game. He takes a sequence of numbers, splits them into two subsequences with the same number of elements and sums the left and right sub-sums, and compares the sub-sums. Please help him.

You are given a number **n** and **2\*n** numbers. Write a program to check whether **the sum of the first n numbers is equal to the sum of the second n numbers**. Print as result “Yes” or “No”. In case of **yes**, print also the sum. In case of **no**, print also the difference between the left and the right sums.

### Input

The input data should be read from the console.

* The first line holds an integer **n** – the count of numbers.
* Each of the next **2\*n** lines holds exactly one number.

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

### Output

* The output must be printed on the console.
* Print “**Yes, sum=*S***” where ***S*** is the sum of the first **n** numbers in case of the sum of the first **n** numbers is equal to the sum of the second **n** numbers.
* Otherwise print “**No, diff=*D***” where ***D*** is the difference between the sum of the first **n** numbers and the sum of the second **n** numbers. ***D*** should always be a **positive number**.

### Constraints

* The number **n** is integer in range [0...500].
* All other numbers are integers in range [-500 000 ... 500 000].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| **4**  3  4  -1  -1  2  1  1  1 | Yes, sum=5 | **3**  1  2  3  1  2  2 | No, diff=1 | **2**  1  1  0  0 | No, diff=2 |

## \* Sunglasses

This problem come from C# Basics practical exam (10 April 2014 Morning). You may submit your solution here: <http://judge.softuni.bg/Contests/2/CSharp-Basics-Exam-10-April-2014-Morning>.

Do you know that the next solar eclipse will occur on April 29, 2014? It will be observable from South Asia, Australia, the Pacific and the Indian Oceans and Antarctica. Spiro is an entrepreneur who wants to cash in on this natural phenomenon. Help him produce protective **sunglasses of different sizes**. You will get 5% of the profit.

### Input

* The input data should be read from the console.
* You have an integer number **N** (always an **odd number**) specifying the height of sunglasses.

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 sunglasses on the console. The sunglasses consist of three parts: **frames**, **lenses** and a **bridge** (the connection between the two frames). Each frame's width is double the height N and should be printed using the character '**\***' (asterisk). Print the lenses with the character '**/**'. Finally, connect the two frames with a bridge that is of size N, using the character '**|**'. Leave the rest of the space between the frames blank.

### Constraints

* The number **N** will be a positive **odd integer** number in range [3…101].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

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

## \*\* Nine-Digit Magic Numbers

This problem come from C# Basics practical exam (10 April 2014 Morning). You may submit your solution here: <http://judge.softuni.bg/Contests/2/CSharp-Basics-Exam-10-April-2014-Morning>.

Petya often plays with numbers. Her recent game was to play with 9-digit numbers and calculate their sums of digits, as well as to split them into triples with special properties. Help her to calculate very special numbers called “***nine-digit magic numbers***”.

You are given two numbers: **diff** and **sum**. Using the **digits from 1 to 7** generate all 9-digit numbers in format **abcdefghi**, such that their sub-numbers **abc**, **def** and **ghi** have a difference **diff** (ghi-def = def-abc = diff), their sum of digits is **sum** and **abc ≤ def** **≤** **ghi**. Numbers holding these properties are also called “***nine-digit magic numbers***”.

Your task is to write a program to print these numbers in increasing order.

### Input

* The input data should be read from the console.
* The number **sum** stays at the first line.
* The number **diff** stays at the second 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 all nine-digit magic numbers matching given difference **diff** and given sum of digits **sum**, in increasing order, each at a separate line. In case no nine-digit magic numbers exits, print “**No**”.

### Constraints

* The number **sum** will be a positive **integer** number in the range [0…100].
* The number **diff** will be a positive **integer** number in the range [0…1000].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 27  46 | 125171217  131177223  221267313 | 1+2+5+1+7+1+2+1+7 = 27; 171-125 = 46; 217-171 = 46  1+3+1+1+7+7+2+2+3 = 27; 177-131 = 46; 223-177 = 46  2+2+1+2+6+7+3+1+3 = 27; 267-221 = 46; 313-267 = 46 |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 24  103 | 121224327  211314417 | 1+2+1+2+2+4+3+2+7 = 24; 224-121 = 103; 327-224 = 103  2+1+1+3+1+4+4+1+7 = 24; 314-211 = 103; 417-314 = 103 |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 12  15 | No | No nine-digit magic numbers with sum=12 and diff=15 |

## \*\* Bits Inverter

This problem come from C# Basics practical exam (10 April 2014 Morning). You may submit your solution here: <http://judge.softuni.bg/Contests/2/CSharp-Basics-Exam-10-April-2014-Morning>.

Once Teodor played the following game: he started from a sequence of white and black balls and flipped the color of the 1st ball, then the color of the 4th ball, then the color of the 7th ball, etc. until the last ball. Flipping was performed by replacing a black b all with a white ball and vice versa. Teodor was a smart mathematician so he wanted to generalize his game in a formal way. This is what he invented.

You are given a **sequence of bytes**. Consider each byte as sequences of exactly 8 bits. You are given also a number **step**. Write a program to invert the bits at positions: **1**, **1 + step**, **1 + 2\*step**, ... Print the output as a sequence of bytes.

Bits in each byte are counted from the leftmost to the rightmost. Bits are numbered starting from 1.

### 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 exactly **n** bytes, each at a separate line and in range [0..255], obtained by applying the bit inversions over the input sequence.

### 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  45  87 | 173  71 | We have the following sequence of 16 bits (2 bytes):  **0**0101101 010**1**0111  We invert the bits 1 and 12 (step=11). We get:  **1**0101101 010**0**0111 |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 3  5  45  87  254 | 169  118  246 | We have the following sequence of 24 bits (3 bytes):  **0**0101**1**01 01**0**1011**1** 1111**1**110  We invert the bits 1, 6, 11, 16 and 21 (step=5). We get:  **1**0101**0**01 01**1**1011**0** 1111**0**110 |

# Homework: Operators Expressions and Statements

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.

## Odd or Even Integers

Write an expression that checks if given integer is **odd or even**. Examples:

|  |  |
| --- | --- |
| **n** | **Odd?** |
| 3 | true |
| 2 | false |
| -2 | false |
| -1 | true |
| 0 | false |

## Gravitation on the Moon

The gravitational field of the Moon is approximately 17% of that on the Earth. Write a program that calculates the **weight of a man on the moon** by a given weight on the Earth. Examples:

|  |  |
| --- | --- |
| **weight** | **weight on the Moon** |
| 86 | 14.62 |
| 74.6 | 12.682 |
| 53.7 | 9.129 |

## Divide by 7 and 5

Write a Boolean expression that checks for given integer if it can be **divided** (without remainder) **by 7 and 5 in the same time**. Examples:

|  |  |
| --- | --- |
| **n** | **Divided by 7 and 5?** |
| 3 | false |
| 0 | false |
| 5 | false |
| 7 | false |
| 35 | true |
| 140 | true |

## Rectangles

Write an expression that calculates **rectangle’s perimeter** and **area** by given **width** and **height**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **width** | **height** | **perimeter** | **area** |
| 3 | 4 | 14 | 12 |
| 2.5 | 3 | 11 | 7.5 |
| 5 | 5 | 20 | 25 |

## Third Digit is 7?

Write **an expression** that checks for given integer **if its third digit** from right-to-left **is 7**. Examples:

|  |  |
| --- | --- |
| **n** | **Third digit 7?** |
| 5 | false |
| **7**01 | true |
| 9**7**03 | true |
| **8**77 | false |
| 777**8**77 | false |
| 9999**7**99 | true |

## Four-Digit Number

Write a program that takes as input a **four-digit number** in format **abcd** (e.g. 2011) and performs the following:

* Calculates the sum of the digits (in our example 2+0+1+1 = 4).
* Prints on the console the number in reversed order: dcba (in our example 1102).
* Puts the last digit in the first position: dabc (in our example 1201).
* Exchanges the second and the third digits: acbd (in our example 2101).

The number has always exactly **4 digits** and cannot start with 0. Examples:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **n** | **sum of digits** | **reversed** | **last digit in front** | **second and third digits exchanged** |
| 2011 | 4 | 1102 | 1201 | 2101 |
| 3333 | 12 | 3333 | 3333 | 3333 |
| 9876 | 30 | 6789 | 6987 | 9786 |

## Point in a Circle

Write **an expression** that checks if given point (**x**, **y**) is inside a **circle K**({**0**, **0**}, **2**). Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **x** | **y** | **inside** |  |
| 0 | 1 | true |
| -2 | 0 | true |
| -1 | 2 | false |
| 1.5 | -1 | true |
| -1.5 | -1.5 | false |
| 100 | -30 | false |
| 0 | 0 | true |
| 0.2 | -0.8 | true |
| 0.9 | -1.93 | false |
| 1 | 1.655 | true |

## Prime Number Check

Write an **expression** that checks if given positive integer number **n** (**n** ≤ 100) is [**prime**](https://en.wikipedia.org/wiki/Prime_number) (i.e. it is divisible without remainder only to itself and 1). Examples:

|  |  |
| --- | --- |
| **n** | **Prime?** |
| 1 | false |
| 2 | true |
| 3 | true |
| 4 | false |
| 9 | false |
| 97 | true |
| 51 | false |
| -3 | false |
| 0 | false |

## Trapezoids

Write an expression that calculates **trapezoid's area** by given sides **a** and **b** and height **h**. Examples:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **a** | **b** | **h** | **area** |  |
| 5 | 7 | 12 | 72 |
| 2 | 1 | 33 | 49.5 |
| 8.5 | 4.3 | 2.7 | 17.28 |
| 100 | 200 | 300 | 45000 |
| 0.222 | 0.333 | 0.555 | 0.1540125 |

## Point Inside a Circle & Outside of a Rectangle

Write an expression that checks for given point (x, y) if it is **within the circle K**({1, 1}, 1.5) and **out of the rectangle R**(top=**1**, left=**-1**, width=**6**, height=**2**). Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **x** | **y** | **inside K & outside of R** |  |
| 1 | 2 | yes |
| 2.5 | 2 | no |
| 0 | 1 | no |
| 2.5 | 1 | no |
| 2 | 0 | no |
| 4 | 0 | no |
| 2.5 | 1.5 | no |
| 2 | 1.5 | yes |
| 1 | 2.5 | yes |
| -100 | -100 | no |

## Bitwise: Extract Bit #3

Using bitwise operators, write an **expression** for finding the value of the bit #**3** of a given unsigned integer. The bits are counted from right to left, starting from bit #0. The result of the expression should be either **1 or 0**. Examples:

|  |  |  |
| --- | --- | --- |
| **n** | **binary representation** | **bit #3** |
| 5 | 00000000 0000**0**101 | 0 |
| 0 | 00000000 0000**0**000 | 0 |
| 15 | 00000000 0000**1**111 | 1 |
| 5343 | 00010100 1101**1**111 | 1 |
| 62241 | 11110011 0010**0**001 | 0 |

## Extract Bit from Integer

Write an expression that extracts from given integer **n** the value of given **bit at index** **p**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **binary representation** | **p** | **bit @ p** |
| 5 | 00000000 00000**1**01 | 2 | 1 |
| 0 | 000000**0**0 00000000 | 9 | 0 |
| 15 | 00000000 000011**1**1 | 1 | 1 |
| 5343 | 00010100 **1**1011111 | 7 | 1 |
| 62241 | 1111**0**011 00100001 | 11 | 0 |

## Check a Bit at Given Position

Write a **Boolean expression** that returns if the **bit at position p** (counting from **0**, starting from the right) in given integer number **n** has value of **1**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **binary representation of n** | **p** | **bit @ p == 1** |
| 5 | 00000000 00000**1**01 | 2 | true |
| 0 | 000000**0**0 00000000 | 9 | false |
| 15 | 00000000 000011**1**1 | 1 | true |
| 5343 | 00010100 **1**1011111 | 7 | true |
| 62241 | 1111**0**011 00100001 | 11 | false |

## Modify a Bit at Given Position

We are given an integer number **n**, a bit value **v** (v=0 or 1) and a position **p**. Write a **sequence of operators** (a few lines of C# code) that modifies **n** to hold the value **v** at the position **p** from the binary representation of **n** while preserving all other bits in **n**. Examples:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **n** | **binary representation of n** | **p** | **v** | **binary result** | **result** |
| 5 | 00000000 00000**1**01 | 2 | 0 | 00000000 00000**0**01 | 1 |
| 0 | 000000**0**0 00000000 | 9 | 1 | 000000**1**0 00000000 | 512 |
| 15 | 00000000 000011**1**1 | 1 | 1 | 00000000 000011**1**1 | 15 |
| 5343 | 00010100 **1**1011111 | 7 | 0 | 00010100 **0**1011111 | 5215 |
| 62241 | 1111**0**011 00100001 | 11 | 0 | 1111**0**011 00100001 | 62241 |

## \* Bits Exchange

Write a program that **exchanges bits** **3**, **4** and **5** with bits **24**, **25** and **26** of **given 32-bit unsigned integer**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **binary representation of n** | **binary result** | **result** |
| 1140867093 | 01000**100** 00000000 01000000 00**010**101 | 01000**010** 00000000 01000000 00**100**101 | 1107312677 |
| 255406592 | 00001**111** 00111001 00110010 00**000**000 | 00001**000** 00111001 00110010 00**111**000 | 137966136 |
| 4294901775 | 11111**111** 11111111 00000000 00**001**111 | 11111**001** 11111111 00000000 00**111**111 | 4194238527 |
| 5351 | 00000**000** 00000000 00010100 11**100**111 | 00000**100** 00000000 00010100 11**000**111 | 67114183 |
| 2369124121 | 10001**101** 00110101 11110111 00**011**001 | 10001**011** 00110101 11110111 00**101**001 | 2335569705 |

## \*\* Bit Exchange (Advanced)

Write a program that **exchanges bits** **{p, p+1, …, p+k-1}** with bits **{q, q+1, …, q+k-1}** of a given 32-bit unsigned integer. The first and the second sequence of bits may **not overlap**. Examples:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **n** | **p** | **q** | **k** | **binary representation of n** | **binary result** | **result** |
| 1140867093 | 3 | 24 | 3 | 01000**100** 00000000 01000000 00**010**101 | 01000**010** 00000000 01000000 00**100**101 | 1107312677 |
| 4294901775 | 24 | 3 | 3 | 11111**111** 11111111 00000000 00**001**111 | 11111**001** 11111111 00000000 00**111**111 | 4194238527 |
| 2369124121 | 2 | 22 | 10 | **10001101 00**110101 1111**0111 000110**01 | **01110001 10**110101 1111**1000 110100**01 | 1907751121 |
| 987654321 | 2 | 8 | 11 | - | - | overlapping |
| 123456789 | 26 | 0 | 7 | - | - | out of range |
| 33333333333 | -1 | 0 | 33 | - | - | out of range |

# 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.

## \*\*– Volleyball

This problem is from Variant 2 of C# Basics exam from 10-04-2014 Evening. You can test your solution [here](http://judge.softuni.bg/Contests/Practice/Index/3#0) .

Vladi loves a lot to play volleyball. However, he is a programmer now and he is very busy. Now he is able to **play only in the holidays** and in the **weekends**. Vladi plays in **2/3 of the holidays** and each **Saturday**, but **not every weekend** – only when he is **not at work** and only when he is **not going to his hometown**. Vladi goes at his hometown **h** weekends in the year. The other weekends are considered “**normal**”. Vladi is **not at work in 3/4 of the normal weekends**. When Vladi is at his hometown, he always plays volleyball with his old friends once, at **Sunday**. In addition, if the year is **leap**, Vladi plays volleyball **15% more times** additionally. We assume the year has **exactly 48 weekends** suitable for volleyball.

Your task is to write a program that calculates **how many times Vladi plays volleyball** (rounded down to the nearest integer number).

### Input

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

* The string “**leap**” for leap year or “**normal**” for year that is not leap.
* The number **p** – number of **holidays** in the year (which are not Saturday or Sunday).
* The number **h** – number of weekends that Vladi spends in his **hometown**.

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 an integer representing how many times Vladi plays volleyball for a year.

### Constraints

* The numbers **p** is in range [0...300] and **h** is in range [0…48].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| leap  5  2 | 45 | 48 weekends total in the year, split into:   * 2 hometown weekends 🡪 2 Sundays 🡪 **2** plays * 46 normal weekends 🡪 46 \* 3 / 4 🡪 **34.5** plays   5 holidays 🡪 5 \* 2/3 🡪 **3.33** plays  Leap year 🡪 additional 15% \* 39.83 🡪 **5.97** plays  Total plays = **45.8** plays 🡪 **45** (rounded down) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| normal  3  2 | 38 | leap  2  3 | 43 | normal  11  6 | 44 | leap  0  1 | 41 | normal  6  13 | 43 |

## \*\* – Odd / Even Sum

This problem is from Variant 2 of C# Basics exam from 10-04-2014 Evening. You can test your solution [here](http://judge.softuni.bg/Contests/Practice/Index/3#1) .

You are given a number **n** and **2\*n** numbers. Write a program to check whether **the sum of the odd numbers is equal to the sum of the even n numbers**. The first number is considered odd, the next even, the next odd again, etc. Print as result “Yes” or “No”. In case of **yes**, print also the sum. In case of **no**, print also the difference between the odd and the even sums.

### Input

The input data should be read from the console.

* The first line holds an integer **n** – the count of numbers.
* Each of the next **2\*n** lines holds exactly one number.

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

### Output

* The output must be printed on the console.
* Print “**Yes, sum=*S***” where ***S*** is the sum of the odd **n** numbers in case of the sum of the odd **n** numbers is equal to the sum of the even **n** numbers.
* Otherwise print “**No, diff=*D***” where ***D*** is the difference between the sum of the odd **n** numbers and the sum of the even **n** numbers. ***D*** should always be a **positive number**.

### Constraints

* The number **n** is integer in range [0...500].
* All other numbers are integers in range [-500 000 ... 500 000].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| **4**  3  **4**  -1  **-1**  2  **1**  1  **1** | Yes, sum=5 | **3**  1  **2**  3  **1**  2  **2** | No, diff=1 | **2**  1  **0**  1  **0** | No, diff=2 |

## \*\* – The Explorer

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/Practice/Index/4#2) .

Bai Vylcho is very an enthusiastic explorer. His passion are the diamonds, he just adores them. Today he is going on an expedition to collect all kind of diamonds, no matter small or large. Help your friend to find all the diamonds in the biggest known cave "The Console Cave". At the only input line you will be given the width of the diamond. The char that forms the outline of the diamonds is '**\***' and the surrounding parts are made of '**-**' (see the examples). Your task is to **print a diamond** of given size **n**.

### Input

Input data should be read from the console.

* The only input line will hold the width of the diamond – **n**.

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.

* The output lines should hold the diamond.

### Constraints

* The number **n is positive odd integer between 3 and 59**, inclusive.
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

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

## \*\* – Bits Up

This problem is from Variant 2 of C# Basics exam from 10-04-2014 Evening. You can test your solution [here](http://judge.softuni.bg/Contests/Practice/Index/3#4) .

You are given a **sequence of bytes**. Consider each byte as sequences of exactly 8 bits. You are given also a number **step**. Write a program to set to 1 the bits at positions: **1**, **1 + step**, **1 + 2\*step**, ... Print the output as a sequence of bytes.

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 exactly **n** bytes, each at a separate line and in range [0..255], obtained by applying the bit inversions over the input sequence.

### 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 | 109  95 | We have the following sequence of 16 bits (2 bytes):  0**1**101101 0101**0**111  We invert the bits 1 and 12 (step=11). We get:  0**1**101101 0101**1**111 |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 3  5  45  87  250 | 111  87  254 | We have the following sequence of 24 bits (3 bytes):  0**0**1011**0**1 010**1**0111 **1**1111**0**10  We invert the bits 1, 6, 11, 16 and 21 (step=5). We get:  0**1**1011**1**1 010**1**0111 **1**1111**1**10 |

## \*\* – Bit Sifting

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/Practice/Index/4#4) .

In this problem we'll be sifting bits through sieves (sift = пресявам, sieve = сито).

You will be given an integer, representing the **bits to sieve**, and several more numbers, representing the **sieves the bits will fall through**. Your task is to follow the bits as they fall down, and determine what comes out of the other end.

|  |  |
| --- | --- |
| ExampleFor this example, imagine we are working with 8-bit integers (the actual problem uses 64-bit ones). Let the initial bits be given as 165 (10100101 in binary), and the sieves be 138 (10001010), 84 (01010100) and 154 (10011010). The 1 bits from the initial number fall through the 0 bits of the sieves and stop if they reach a 1 bit; if they make it to the end, they become a part of the final number.In this case, the final number is 33 (00100001), which has two 1 bits in its binary form – the answer is 2. | 10100101↓ ↓ ↓ ↓10001010↓ ↓ ↓01010100↓ ↓10011010↓ ↓ 00100001 |

### Input

The input data should be read from the console.

* On the first line of input, you will read an integer representing the bits to sieve.
* On the second line of input, you will read an integer N representing the number of sieves.
* On the next N lines of input, you will read N integers representing the sieves.

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

### Output

The output must be printed on the console.

On the single line of the output you must print **the count of "1" bits** in the final result.

### Constraints

* All numbers in the input will be between 0 and 18,446,744,073,709,551,615.
* The count of sieves N is in range [0…100].
* Allowed work time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 584938644408189469  3  1817781288526917737  8601652436058397548  51827709899390606 | 4 | 918045605434484408  0 | 35 | 5019588773529942006  1  5295337384025297044 | 17 |

# 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 laetters**, 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**. |

# Homework: Conditional Statements

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.

## Exchange If Greater

Write an **if**-statement that takes two integer variables a and b and **exchanges** their values if the first one is greater than the second one. As a result print the values a and b, separated by a space. Examples:

|  |  |  |
| --- | --- | --- |
| **a** | **b** | **result** |
| 5 | 2 | 2 5 |
| 3 | 4 | 3 4 |
| 5.5 | 4.5 | 4.5 5.5 |

## Bonus Score

Write a program that applies bonus score to given score in the range [1…9] by the following rules:

* If the score is between 1 and 3, the program multiplies it by 10.
* If the score is between 4 and 6, the program multiplies it by 100.
* If the score is between 7 and 9, the program multiplies it by 1000.
* If the score is 0 or more than 9, the program prints “invalid score”.

Examples:

|  |  |
| --- | --- |
| **score** | **result** |
| 2 | 20 |
| 4 | 400 |
| 9 | 9000 |
| -1 | invalid score |
| 10 | invalid score |

## Check for a Play Card

Classical play cards use the following signs to designate the card face: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K and A. Write a program that enters a string and prints “yes” if it is a valid card sign or “no” otherwise. Examples:

|  |  |
| --- | --- |
| **character** | **Valid card sign?** |
| 5 | yes |
| 1 | no |
| Q | yes |
| q | no |
| P | no |
| 10 | yes |
| 500 | no |

## Multiplication Sign

Write a program that shows the sign (+, - or 0) of the product of three real numbers, without calculating it. Use a sequence of **if** operators. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **c** | **result** |
| 5 | 2 | 2 | + |
| -2 | -2 | 1 | + |
| -2 | 4 | 3 | - |
| 0 | -2.5 | 4 | 0 |
| -1 | -0.5 | -5.1 | - |

## The Biggest of 3 Numbers

Write a program that finds the **biggest of three numbers**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **c** | **biggest** |
| **5** | 2 | 2 | 5 |
| -2 | -2 | **1** | 1 |
| -2 | **4** | 3 | 4 |
| 0 | -2.5 | **5** | 5 |
| **-0.1** | -0.5 | -1.1 | -0.1 |

## The Biggest of Five Numbers

Write a program that finds the **biggest of five numbers** by using only five if statements. Examples:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c** | **d** | **e** | **biggest** |
| **5** | 2 | 2 | 4 | 1 | 5 |
| -2 | -22 | **1** | 0 | 0 | 1 |
| -2 | **4** | 3 | 2 | 0 | 4 |
| 0 | -2.5 | 0 | **5** | **5** | 5 |
| -3 | -0.5 | -1.1 | -2 | **-0.1** | -0.1 |

## Sort 3 Numbers with Nested Ifs

Write a program that enters **3 real numbers** and prints them sorted in descending order. Use nested **if** statements. Don’t use arrays and the built-in sorting functionality. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **c** | **result** |
| 5 | 1 | 2 | 5 2 1 |
| -2 | -2 | 1 | 1 -2 -2 |
| -2 | 4 | 3 | 4 3 -2 |
| 0 | -2.5 | 5 | 5 0 -2.5 |
| -1.1 | -0.5 | -0.1 | -0.1 -0.5 -1.1 |
| 10 | 20 | 30 | 30 20 10 |
| 1 | 1 | 1 | 1 1 1 |

## Digit as Word

Write a program that asks for a **digit** (0-9), and depending on the input, **shows the digit as a word** (in English). Print “not a digit” in case of invalid inut. Use a **switch** statement. Examples:

|  |  |
| --- | --- |
| **d** | **result** |
| 2 | two |
| 1 | one |
| 0 | zero |
| 5 | five |
| -0.1 | not a digit |
| hi | not a digit |
| 9 | nine |
| 10 | not a digit |

## Play with Int, Double and String

Write a program that, depending on the user’s choice, inputs an **int**, **double** or **string** variable. If the variable is **int** or **double**, the program increases it by one. If the variable is a **string**, the program appends "**\***" at the end. Print the result at the console. Use **switch** statement. Example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **program** | **user** |  | **program** | **user** |
| Please choose a type: 1 --> int  2 --> double  3 --> string | 3 | Please choose a type: 1 --> int  2 --> double  3 --> string | 2 |
| Please enter a string: | hello | Please enter a double: | 1.5 |
| hello\* |  | 2.5 |  |

## \* Beer Time

A beer time is after 1:00 PM and before 3:00 AM. Write a program that **enters a time** in format “hh:mm tt” (an hour in range [01...12], a minute in range [00…59] and AM / PM designator) and prints “**beer time**” or “**non-beer time**” according to the definition above or “**invalid time**” if the time cannot be parsed. Note that you may need to learn how to parse dates and times. Examples:

|  |  |
| --- | --- |
| **time** | **result** |
| 1:00 PM | beer time |
| 4:30 PM | beer time |
| 10:57 PM | beer time |
| 8:30 AM | non-beer time |
| 02:59 AM | beer time |
| 03:00 AM | non-beer time |
| 03:26 AM | non-beer time |

## \* Number as Words

Write a program that **converts a number in the range [0…999] to words**, corresponding to the English pronunciation. Examples:

|  |  |
| --- | --- |
| **numbers** | **number as words** |
| 0 | Zero |
| 9 | Nine |
| 10 | Ten |
| 12 | Twelve |
| 19 | Nineteen |
| 25 | Twenty five |
| 98 | Ninety eight |
| 273 | Two hundred and seventy three |
| 400 | Four hundred |
| 501 | Five hundred and one |
| 617 | Six hundred and seventeen |
| 711 | Seven hundred and eleven |
| 999 | Nine hundred and ninety nine |

## \* Zero Subset

We are given 5 integer numbers. Write a program that finds all **subsets of these numbers whose sum is 0**. Assume that repeating the same subset several times is not a problem. Examples:

|  |  |
| --- | --- |
| **numbers** | **result** |
| 3 -2 1 1 8 | -2 + 1 + 1 = 0 |
| 3 1 -7 35 22 | no zero subset |
| 1 3 -4 -2 -1 | 1 + -1 = 0  1 + 3 + -4 = 0  3 + -2 + -1 = 0 |
| 1 1 1 -1 -1 | 1 + -1 = 0  1 + 1 + -1 + -1 = 0  1 + -1 + 1 + -1 = 0  … |
| 0 0 0 0 0 | 0 + 0 + 0 + 0 + 0 = 0 |

Hint: you may check for zero each of the 32 subsets with 32 if statements.

# Exam problems.\*\*

**All of the problems below are given from Variant 5 of C# Basics Practical Exam (12 April 2014 Morning). You can submit your solutions** [**HERE**](http://judge.softuni.bg/Contests/6/CSharp-Basics-Exam-12-April-2014-Morning)**.**

**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.

## \* – Triangle

You are given a two-dimensional Cartesian coordinate system and three points A, B, C with coordinates: A(Ax, Ay), B(Bx, By), C(Cx, Cy). Write a program to check if these **three points can form a triangle**. Then calculate the area of this triangle. To find the distance between two points with the coordinates (x1, y1) and (x2, y2) use the formula:

You can use the inequalities of a triangle to check whether three segments **a**, **b** and **c** can form a triangle:

; ;

To calculate the area you can use Heron`s formula (a method for calculating the area of a triangle when you know the lengths of all three sides). Let **a**, **b**, **c** be the lengths of the sides of a triangle. Thus:

, where **p** is half the perimeter: .

### Input

The input data comes from the console. It consists of exactly 6 lines holding the coordinates of the three points: **Ax**-coordinate, **Ay**-coordinate, **Bx**-coordinate, **By**-coordinate, **Cx**-coordinate and **Cy**-coordinate. 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 and must contain two lines.

* **First line:** If the given points can form a triangle you must print the message “**Yes**”, otherwise “**No**”.
* **Second line:** If the given points can form a triangle you must print the **area of the triangle** rounded to two decimal places (see the examples), otherwise you must print the **distance between point A and point B**. Use "**.**" as decimal separator.

### Constraints

* The coordinate **X** is integer in the range [-10000… 10000] inclusive.
* The coordinate **Y** is integer in the range [-10000… 10000] inclusive.
* Allowed work time for your program: 0.1 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** | **Comments** |  | **Input** | **Output** | **Comments** |
| 2  2  0  0  1  1 | No  2.83 |  |  | 2  3  0  -1  4  -2 | Yes  9.00 |  |

## \* – Pairs

You are given **2\*N elements** (even number of integer numbers). The first and the second element form a **pair**, the third and the fourth element form a pair as well, etc. Each pair has a **value**, calculated as the **sum** of its two elements. Your task is to write a program to check **whether all pairs have the same value** or print the **max difference** between two consecutive values.

### Input

You are given at the console **even number of integers**, all in a single line, separated by a space.

### Output

The output is single line, printed at the console.

* In case all pairs have the same value, print "**Yes, value=…**" and the value.
* Otherwise, print "**No, maxdiff=…**" and the maximal difference between two consecutive values, always a **positive integer**.

### Constraints

* All input values will be integers in the range [-1000…1000] inclusive.
* The count of elements is even number in the range [2…1000] inclusive.
* Allowed work time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1 2 0 3 4 -1 | Yes, value=3 | values = {3, 3, 3} --> equal values |
| 1 2 2 2 | No, maxdiff=1 | values = {3, 4}, different values --> max difference = 4-3 = 1 |
| 1 1 3 1 2 2 0 0 | No, maxdiff=4 | values = {2, 4, 4, 0}, differences = {2, 0, 4}, max difference = 4 |
| 5 5 | Yes, value=10 | values = {10} --> single value --> equal values |
| -1 0 0 -1 | Yes, value=-1 | values = {-1, -1}, equal values |

## \* – House

Ivaylo decided he needs a new house. Since he is not a structural engineer yet, you have to help him construct the building from the ground zero.

The roof is a triangle. The walls are straight vertical lines. The base is a straight horizontal line. The roof, the walls and the base of the house it build of '**\***'. Everything else is filled with '.' (see the examples below to catch the idea).

You will be given an odd integer **N**, representing the width and the height of the house. The roof’s top starts from the center (**N+1)/2** and forms a triangle with base of width **N**. The roof’s height is (**N+1)/2**. The distance between the roofs’ end point and the walls of the building is **N/4**, rounded down to an integer number. See the examples below to understand better these formulas.

### Input

* Input data is read from the console.
* The number **N** stays alone 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 data must be printed on the console.
* You must print at the console a house of size **N** following the formulas above and the examples below.

### Constraints

* **N** will be an **odd** number between **5** and **49**.
* Time limit: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 5 | ..\*..  .\*.\*.  \*\*\*\*\*  .\*.\*.  .\*\*\*. |  | 7 | ...\*...  ..\*.\*..  .\*...\*.  \*\*\*\*\*\*\*  .\*...\*.  .\*...\*.  .\*\*\*\*\*. | 9 | ....\*....  ...\*.\*...  ..\*...\*..  .\*.....\*.  \*\*\*\*\*\*\*\*\*  ..\*...\*..  ..\*...\*..  ..\*...\*..  ..\*\*\*\*\*.. |

## \*\* – Magic Dates

Consider we are given a **date** in format dd-mm-yyyy, e.g. 17-03-2007. We calculate the **weight of this date** by joining together all its digits, multiplying each digit by each of the other digits and finally sum all obtained products. In our case we will have 8-digits: 17032007, so the weight is 1\*7 + 1\*0 + 1\*3 + 1\*2 + 1\*0 + 1\*0 + 1\*7 + 7\*0 + 7\*3 + 7\*2 + 7\*0 + 7\*0 + 7\*7 + 0\*3 + 0\*2 + 0\*0 + 0\*0 + 0\*7 + 3\*2 + 3\*0 + 3\*0 + 3\*7 + 2\*0 + 2\*0 + 2\*7 + 0\*0 + 0\*7 + 0\*7 = 144.

Your task is to write a program that finds all **magic dates**: **dates between two fixed years matching given magic weight**. The dates should be printed in increasing order in format dd-mm-yyyy. We use the traditional calendar (years have 12 months, each having 28, 29, 30 or 31 days, etc.). Years start from 1 January and end in 31 December.

### Input

The input data should be read from the console. It consists of 3 lines:

* The first line holds an integer number – **start year**.
* The second line holds an integer number – **end year**.
* The third line holds an integer number – **magic weight**.

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 dates in **format dd-mm-yyyy** in **alphabetical order**. Each string should stay on a separate line. In case no magic dates exist, print “**No**”.

Beware that the **regional settings** at your computer and at the judge's computer may be different!

### Constraints

* The **start** and **end year** are **integers** in the range [1900-2100].
* The **magic weight** is an integer number in range [1…1000].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 2007  2007  144 | 17-03-2007  13-07-2007  31-07-2007 | 2003  2004  1500 | No | 2012  2014  80 | 09-01-2013  17-01-2013  23-03-2013  11-07-2013  01-09-2013  10-09-2013  09-10-2013  17-10-2013  07-11-2013  24-11-2013  14-12-2013  23-11-2014  13-12-2014  31-12-2014 | 2011  2012  14 | 01-01-2011  10-01-2011  01-10-2011  10-10-2011 |

## \*\* – Bit Killer

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 remove (kill) 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 | 90  188 | We have the following input sequence of 16 bits (2 bytes):  0**1**101101 0101**0**111. We kill the bits 1 and 12 (step=11). Obtained sequence: 01011010 101111. Padding: 2 zeroes at the end. Result: **01011010 10111100**. |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 3  2  45  87  250 | 97  240 | 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 kill bits 1, 3, …, 23 (step=2). Obtain the sequence: 01100001 1111. We pad it with 4 zeroes at the end to obtain 2 full bytes. Result: **01100001 11110000**. |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 2  2  45  87 | 97 | 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 kill bits 1, 3, 5, …, 15 (step=2). Obtained sequence: 01100001 (8 bits). No padding is needed. Result: **01100001**. |

# Homework: Loops

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.

## Numbers from 1 to N

Write a program that enters from the console a positive integer **n** and **prints all the numbers from 1 to n**, on a single line, separated by a space. Examples:

|  |  |
| --- | --- |
| **n** | **output** |
| 3 | 1 2 3 |
| 5 | 1 2 3 4 5 |

## Numbers Not Divisible by 3 and 7

Write a program that enters from the console a positive integer **n** and prints all the **numbers from 1 to n not divisible by 3 and 7**, on a single line, separated by a space. Examples:

|  |  |
| --- | --- |
| **n** | **output** |
| 3 | 1 2 |
| 10 | 1 2 4 5 8 10 |

## Min, Max, Sum and Average of N Numbers

Write a program that reads from the console a sequence of **n** integer numbers and returns the **minimal**, the **maximal** number, the sum and the average of all numbers (displayed with 2 digits after the decimal point). The **input** starts by the number **n** (alone in a line) followed by **n lines**, each holding an integer number. The **output** is like in the examples below. Examples:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **input** | **output** |  | **input** | **output** |
| 3  2  5  1 | min = 1  max = 5  sum = 8  avg = 2.67 | 2  -1  4 | min = -1  max = 4  sum = 3  avg = 1.50 |

## Print a Deck of 52 Cards

Write a program that generates and prints **all possible cards from a** [**standard deck of 52 cards**](http://en.wikipedia.org/wiki/Standard_52-card_deck) (without the jokers). The cards should be printed using the classical notation (like 5♠, A♥, 9♣ and K♦). The card faces should start from 2 to A. Print each card face in its four possible suits: clubs, diamonds, hearts and spades. Use 2 nested for-loops and a switch-case statement.

|  |
| --- |
| **output** |
| 2♣ 2♦ 2♥ 2♠  3♣ 3♦ 3♥ 3♠  …  K♣ K♦ K♥ K♠  A♣ A♦ A♥ A♠ |

## Calculate 1 + 1!/X + 2!/X2 + … + N!/XN

Write a program that, for a given two integer numbers **n** and **x**, calculates the sum S = 1 + 1!/x + 2!/x2 + … + n!/xn. Use only one loop. Print the result with 5 digits after the decimal point.

|  |  |  |
| --- | --- | --- |
| **n** | **x** | **S** |
| 3 | 2 | 2.75000 |
| 4 | 3 | 2.07407 |
| 5 | -4 | 0.75781 |

## Calculate N! / K!

Write a program that calculates **n! / k!** for given **n** and **k** (1 < **k** < **n** < 100). Use only one loop. Examples:

|  |  |  |
| --- | --- | --- |
| **n** | **k** | **n! / k!** |
| 5 | 2 | 60 |
| 6 | 5 | 6 |
| 8 | 3 | 6720 |

## Calculate N! / (K! \* (N-K)!)

In combinatorics, the number of ways to choose **k** different members out of a group of **n** different elements (also known as the number of [**combinations**](http://en.wikipedia.org/wiki/Combination)) is calculated by the following formula:

 \binom nk = \frac{n!}{k!(n-k)!},

For example, there are 2598960 ways to withdraw 5 cards out of a standard deck of 52 cards. Your task is to write a program that calculates **n! / (k! \* (n-k)!)** for given **n** and **k** (1 < **k** < **n** < 100). Try to use only two loops. Examples:

|  |  |  |
| --- | --- | --- |
| **n** | **k** | **n! / (k! \* (n-k)!)** |
| 3 | 2 | 3 |
| 4 | 2 | 6 |
| 10 | 6 | 210 |
| 52 | 5 | 2598960 |

## Catalan Numbers

In combinatorics, the [Catalan numbers](http://en.wikipedia.org/wiki/Catalan_number) are calculated by the following formula:

C_n = \frac{1}{n+1}{2n\choose n} = \frac{(2n)!}{(n+1)!\,n!} = \prod\limits_{k=2}^{n}\frac{n+k}{k} \qquad\mbox{ for }n\ge 0.

Write a program to calculate the **nth Catalan number** by given **n** (1 < n < 100). Examples:

|  |  |
| --- | --- |
| **n** | **Catalan(n)** |
| 0 | 1 |
| 5 | 42 |
| 10 | 16796 |
| 15 | 9694845 |

## Matrix of Numbers

Write a program that reads from the console a positive integer number n (1 ≤ n ≤ 20) and **prints a matrix** like in the examples below. Use two nested loops. Examples:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **n** | **matrix** |  | **n** | **matrix** |  | **n** | **matrix** |
| 2 | 1 2  2 3 | 3 | 1 2 3  2 3 4  3 4 5 | 4 | 1 2 3 4  2 3 4 5  3 4 5 6  4 5 6 7 |

## Odd and Even Product

You are given **n** integers (given in a single line, separated by a space). Write a program that checks whether the product of the odd elements is equal to the product of the even elements. Elements are counted from 1 to n, so the first element is odd, the second is even, etc. Examples:

|  |  |
| --- | --- |
| **numbers** | **result** |
| **2** 1 **1** 6 **3** | yes  product = 6 |
| **3** 10 **4** 6 **5** 1 | yes  product = 60 |
| **4** 3 **2** 5 **2** | no  odd\_product = 16  even\_product = 15 |

## Random Numbers in Given Range

Write a program that enters 3 integers n, min and max (min ≤ max) and prints n random numbers in the range [min...max]. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **min** | **max** | **random numbers** |
| 5 | 0 | 1 | 1 0 0 1 1 |
| 10 | 10 | 15 | 12 14 12 15 10 12 14 13 13 11 |

Note that the above output is just an example. Due to randomness, your program most probably will produce different results.

## \* Randomize the Numbers 1…N

Write a program that enters in integer n and prints the numbers 1, 2, …, n in random order. Examples:

|  |  |
| --- | --- |
| **n** | **randomized numbers 1…n** |
| 3 | 2 1 3 |
| 10 | 3 4 8 2 6 7 9 1 10 5 |

Note that the above output is just an example. Due to randomness, your program most probably will produce different results. You might need to use [arrays](http://msdn.microsoft.com/en-us/library/aa288453(v=vs.71).aspx).

## Binary to Decimal Number

Using loops write a program that converts a [binary integer](http://en.wikipedia.org/wiki/Binary_numeral_system) number to its decimal form. The input is entered as string. The output should be a variable of type long. Do not use the built-in .NET functionality. Examples:

|  |  |
| --- | --- |
| **binary** | **decimal** |
| 0 | 0 |
| 11 | 3 |
| 1010101010101011 | 43691 |
| 1110000110000101100101000000 | 236476736 |

## Decimal to Binary Number

Using loops write a program that converts an integer number to its [binary representation](http://en.wikipedia.org/wiki/Binary_numeral_system). The input is entered as long. The output should be a variable of type string. Do not use the built-in .NET functionality. Examples:

|  |  |
| --- | --- |
| **decimal** | **binary** |
| 0 | 0 |
| 3 | 11 |
| 43691 | 1010101010101011 |
| 236476736 | 1110000110000101100101000000 |

## Hexadecimal to Decimal Number

Using loops write a program that converts a [hexadecimal integer](http://en.wikipedia.org/wiki/Hexadecimal) number to its decimal form. The input is entered as string. The output should be a variable of type long. Do not use the built-in .NET functionality. Examples:

|  |  |
| --- | --- |
| **hexadecimal** | **decimal** |
| FE | 254 |
| 1AE3 | 6883 |
| 4ED528CBB4 | 338583669684 |

## Decimal to Hexadecimal Number

Using loops write a program that converts an integer number to its [hexadecimal representation](http://en.wikipedia.org/wiki/Hexadecimal). The input is entered as long. The output should be a variable of type string. Do not use the built-in .NET functionality. Examples:

|  |  |
| --- | --- |
| **decimal** | **hexadecimal** |
| 254 | FE |
| 6883 | 1AE3 |
| 338583669684 | 4ED528CBB4 |

## \* Calculate GCD

Write a program that calculates the [**greatest common divisor**](http://en.wikipedia.org/wiki/Greatest_common_divisor) (**GCD**) of given two integers a and b. Use the **Euclidean algorithm** (find it in Internet). Examples:

|  |  |  |
| --- | --- | --- |
| **a** | **b** | **GCD(a, b)** |
| 3 | 2 | 1 |
| 60 | 40 | 20 |
| 5 | -15 | 5 |

## \* Trailing Zeroes in N!

Write a program that calculates with how many zeroes the factorial of a given number n has at its end. Your program should work well for very big numbers, e.g. n=100000. Examples:

|  |  |  |
| --- | --- | --- |
| **n** | **trailing zeroes of n!** | **explaination** |
| 10 | 2 | 36288**00** |
| 20 | 4 | 243290200817664**0000** |
| 100000 | 24999 | think why |

## \*\* Spiral Matrix

Write a program that reads from the console a positive integer number n (1 ≤ n ≤ 20) and **prints a matrix** holding the numbers from 1 to n\*n in the form of **square spiral** like in the examples below. Examples:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **n** | **matrix** |  | **n** | **matrix** |  | **n** | **matrix** |
| 2 | 1 2  4 3 | 3 | 1 2 3  8 9 4  7 6 5 | 4 | 1 2 3 4  12 13 14 5  11 16 15 6  10 9 8 7 |

# Exam problems.\*\*

**All of the problems below are given from Variant 6 of C# Basics Practical Exam (12 April 2014 Evening). You can submit your solutions** [**HERE**](http://judge.softuni.bg/Contests/7/CSharp-Basics-Exam-12-April-2014-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://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.

## \*\* – Exam Schedule

At SoftUni we have a new trainee Stamat, who is assigned to make **schedules for the entrance exams**. Since today is his first day at work he is a little bit nervous and he is not working very fast. Unfortunately, it seems that he will not have enough time to make the schedule for the next exam and there is no one else to do the job … except you of course. You will be given **exam starting time** in the standard 12-hour clock (**hours**, **minutes** and **part of the day**) and **exam duration** (**hours** and **minutes**). Your job is to write a program that calculates **at what time the exam ends**.

\* Note that the **standard 12-hours clock** uses the following arrangements of the hours of the day: 12AM (midnight), 1AM, 2AM, 3AM, 4AM, 5AM, 6AM, 7AM, 8AM, 9AM, 10AM, 11AM, 12PM (noon), 1PM, 2PM, 3PM, 4PM, 5PM, 6PM, 7PM, 8PM, 9PM, 10PM, 11PM, 12AM, 1AM, … (learn more at <http://en.wikipedia.org/wiki/12-hour_clock>).

### Input

The input data should be read from the console. The input data consists of exactly 5 lines:

* The first three lines are holding the exam start time: **hour**, **minutes** and **part of the day (AM or PM)**.
* The last two lines are holding two integer number: the exam **duration hours** and **minutes**.

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

### Output

You have to print the time the exam ends in format **HH:MM:PartOfDay**.

### Constraints

* The **starting hour** will be an integer number in the range [1…12] inclusive.
* The **starting minutes** will an integer number in the range [0…59] inclusive.
* The **part of the day** will consist of exactly two capital letters: **AM** or **PM**.
* The **duration hours** will be an integer number between [0…23] inclusive.
* The **duration minutes** will be an integer number between [0…59] inclusive.
* Allowed work time for your program: 0.1 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 9  30  AM  6  00 | 03:30:PM | 2  0  PM  2  30 | 04:30:PM |  | 11  30  AM  2  0 | 01:30:PM |  | 11  59  PM  0  3 | 12:02:AM |

## \*\* – Odd / Even Elements

You are given **N numbers**. Calculate the **sum**, **min** and **max** of its **odd elements** and **sum**, **min** and **max** of its **even elements.** Consider that the first element is odd, the second is even, etc.

### Input

The input data should be read from the console. It will consists of exactly one line.

* At the first line **N numbers** will be given, separated one from another 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

You have to print the output in a single line at the console in the following format:

* **OddSum=…, OddMin=…, OddMax=…, EvenSum=…, EvenMin=…, EvenMax=…**

Print the numbers in the output without any unneeded trailing zeroes (i.e. print 1.5 instead of 1.50; 1 instead of 1.00). In case the sum, the minimal or the maximal element cannot be calculated (due to missing data), print "**No**".

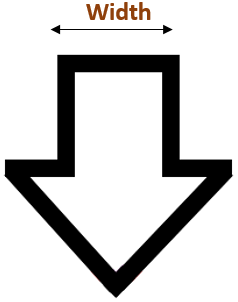
### Constraints

* All numbers in the input will be in the range [-1 000 000 … 1 000 000], with no more than 10 digits (total, before and after the decimal point). The decimal separator in the non-integer numbers will be '**.**' and the numbers will have up to 2 digits after the decimal separator.
* The count N of the numbers in the input is in the range [0 … 1000].
* All numbers in the output should be formatted **without unneded trailing zeroes**.
* Allowed work time for your program: 0.1 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 3 5 4 2 1 | OddSum=9, OddMin=2, OddMax=5, EvenSum=8, EvenMin=1, EvenMax=4 |
| 3 -2 8 11 -3 | OddSum=8, OddMin=-3, OddMax=8, EvenSum=9, EvenMin=-2, EvenMax=11 |
| 1 | OddSum=1, OddMin=1, OddMax=1, EvenSum=No, EvenMin=No, EvenMax=No |
| 1.5 -2.5 | OddSum=1.5, OddMin=1.5, OddMax=1.5, EvenSum=-2.5, EvenMin=-2.5, EvenMax=-2.5 |
| 1.5 1.75 1.5 1.75 | OddSum=3, OddMin=1.5, OddMax=1.5, EvenSum=3.5, EvenMin=1.75, EvenMax=1.75 |

## \*\* – Arrow

SoftUni has opened a new training center in Kaspichan, but the people there did not know how to find it. Your task is to **print a vertical arrow**, which will be used to indicate the path to the new building in the city. This will help thousands of people to become software engineers. Please help them!

### Input

The input data should be read from the console.

* On the only line will hold and integer number **N** (always **odd** number), indicating the **width** of the arrow.

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. Use the “**#**” (number sign) to mark the arrow and “**.**” (dot) for the rest. Follow the examples below.

### Constraints

* **N** will always be a positive **odd** number between **3** and **79** inclusive.
* Allowed working time for your program: 0.1 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 5 | ..#####..  ..#...#..  ..#...#..  ..#...#..  ###...###  .#.....#.  ..#...#..  ...#.#...  ....#.... | 9 | ....#########....  ....#.......#....  ....#.......#....  ....#.......#....  ....#.......#....  ....#.......#....  ....#.......#....  ....#.......#....  #####.......#####  .#.............#.  ..#...........#..  ...#.........#...  ....#.......#....  .....#.....#.....  ......#...#......  .......#.#.......  ........#........ |  | 3 | .###.  .#.#.  ##.##  .#.#.  ..#.. |

## \*\* – Five Special Letters

We are given two numbers: **start** and **end**. Write a program to **generate all sequences of 5 letters**, each from the set { '**a**', '**b**', '**c**', '**d**', '**e**' }, such that the weight of these 5 letters is a number in the range [**start** … **end**] inclusively. Print them in alphabetical order, in a single line, separated by a space.

The **weight of a single letter** is calculated as follows: weight('**a**') = **5**; weight('**b**') = **-12**; weight('**c**') = **47**; weight('**d**') = **7**; weight('**e**') = **-32**. The **weight of a sequence** of letters c1c2…cn is the calculated by first removing all repeating letters (from right to left) and then calculate the formula:

weight(c1c2…cn) = 1\*weight(c1) + 2\*weight(c2) + … + n\*weight(cn)

For example, the weight of "**bcddc**" is calculated as follows: First we remove the repeating letters and we get "**bcd**". Then we apply the formula: 1\*weight('**b**') + 2\*weight('**c**') + 3\*weight('**d**') = 1\*(-12) + 2\*47 + 3\*7 = 103. Another example: weight("cadea") = weight("cade") = 1\*47 + 2\*5 + 3\*7 - 4\*32 = -50.

### Input

The input data should be read from the console. It will consist of 2 lines:

* The number **start** stays at the first line.
* The number **end** stays at the second 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 be separated than the next string by a single space. In case no 5-letter strings exist with a weight in the specified range, print “**No**”.

### Constraints

* The numbers **start** and **end** will be an **integers** in the range [-10000…10000].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 40  42 | bcead bdcea | weight("bcead") = 41  weight("bdcea") = 40 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| -1  1 | bcdea cebda eaaad eaada eaadd eaade eaaed eadaa eadad eadae eadda eaddd eadde eadea eaded eadee eaead eaeda eaedd eaede eaeed eeaad eeada eeadd eeade eeaed eeea | 200  300 | baadc babdc badac badbc badca badcb badcc badcd baddc bbadc bbdac bdaac bdabc bdaca bdacb bdacc bdacd bdadc bdbac bddac beadc bedac eabdc ebadc ebdac edbac | 300  400 | No |

## \*\* – Bit Roller

Nakov enjoys playing with bits very much. Yesterday he invented a new game. He takes a 19-bit number and rolls it on the right (moves its most right bit at the left most position). He tried this several times and he found it is too easy. Then he invented a more complex game: freeze a certain bit as a pillar and roll right all other bits several times. Now he is happy but he wants to automate this process.

Help Nakov to write a program that **rolls r times a 19-bit number n with a frozen bit at position f**.

Example: we have the number **n** = **2521**, which is **0000000100111011001** in binary (as a 19-bit number). We **freeze** the bit at position **f** = **8** (we count the positions from the right, starting from 0). We roll out the number **r** = **4** times. We obtain the result **295245** as follows:

* 2521(10) = 0000000100**1**11011001 🡪 1000000010**1**01101100 (roll right with frozen position 8)
* 1000000010**1**01101100 🡪 0100000001**1**00110110 (roll right with frozen position 8)
* 0100000001**1**00110110 🡪 0010000000**1**10011011 (roll right with frozen position 8)
* 0010000000**1**10011011 🡪 1001000000**1**01001101 = 295245(10) (roll right with frozen position 8)

### Input

The input data should be read from the console. It will consist of 3 lines:

* The number **n** stays at the first line.
* The number **f** stays at the second line.
* The number **r** stays at the third line.

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

### Output

Print the obtained result after rolling **r** times **n** with a frozen bit at position **f** at the console (as decimal number).

### Constraints

* The number **n** will be a 19-bit unsigned **integer** (in the range [0…524287]).
* The number **f** will be integer in the range [0…18].
* The number **r** will be integer in the range [0…100].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 2521  8  4 | 295245 | 2521(10) = 0000000100**1**11011001 🡪 1000000010**1**01101100 🡪 0100000001**1**00110110 🡪 0010000000**1**10011011 🡪 1001000000**1**01001101 = 295245(10) |
| 480678  18  2 | 447849 | 480679(10) = **1**110101010110100110 🡪 **1**011010101011010011 🡪 **1**101101010101101001 = 447849(10) |

# Homework: C# Advanced Topics

This document defines the homework assignments from the [“C# Basics“ Course @ Software University](http://softuni.bg/courses/csharp-basics/).

## Fibonacci Numbers

Define a method Fib(n) that calculates the nth [Fibonacci number](https://en.wikipedia.org/wiki/Fibonacci_number). Examples:

|  |  |
| --- | --- |
| **n** | **Fib(n)** |
| 0 | 1 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 5 |
| 5 | 8 |
| 6 | 13 |
| 11 | 144 |
| 25 | 121393 |

## Prime Checker

Write a Boolean method IsPrime(n) that check whether a given integer number n is [prime](https://en.wikipedia.org/wiki/Prime_number). Examples:

|  |  |
| --- | --- |
| **n** | **IsPrime(n)** |
| 0 | false |
| 1 | false |
| 2 | true |
| 3 | true |
| 4 | false |
| 5 | true |
| 323 | false |
| 337 | true |
| 6737626471 | true |
| 117342557809 | false |

## Primes in Given Range

Write a method that calculates **all prime numbers in given range** and returns them as list of integers:

|  |
| --- |
| static List<int> FindPrimesInRange(startNum, endNum)  {  …  } |

Write a method to **print a list of integers**. Write a program that enters two integer numbers (each at a separate line) and prints all primes in their range, separated by a comma.

Examples:

|  |  |
| --- | --- |
| **Start number End number** | **Output** |
| 0  10 | 2, 3, 5, 7 |
| 5  11 | 5, 7, 11 |
| 100 200 | 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199 |
| 250  950 | 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947 |
| 100  50 | *(empty list)* |

## Difference between Dates

Write a program that enters two dates in format dd.MM.yyyy and returns the number of days between them. Examples:

|  |  |
| --- | --- |
| **First date Second date** | **Days between** |
| 17.03.2014  30.04.2014 | 44 |
| 17.03.2014  17.03.2014 | 0 |
| 14.06.1980  5.03.2014 | 12317 |
| 5.03.2014  3.03.2014 | -2 |

## Sorting Numbers

Write a program that reads a number n and a sequence of n integers, sorts them and prints them. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| **5**  3  -3  2  122  0 | -3  0  2  3  122 |
| **3**  0  1  0 | 0  0  1 |

## Longest Area in Array

Write a program to find the **longest area of equal elements** in array of strings. You first should read an integer n and n strings (each at a separate line), then find and print the longest sequence of equal elements (first its length, then its elements). If multiple sequences have the same maximal length, print the leftmost of them. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 6  hi  hi  hello  ok  ok  ok | 3  ok  ok  ok |
| 2  SoftUni  Hello | 1  SoftUni |
| 4  hi  hi  hi  hi | 4  hi  hi  hi  hi |
| 5  wow  hi  hi  ok  ok | 2  hi  hi |

## Matrix of Palindromes

Write a program to generate the following matrix of palindromes of **3** letters with r rows and c columns:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 6 | aaa aba aca ada aea afa  bbb bcb bdb beb bfb bgb  ccc cec cdc cfc cgc chc |
| 2 3 | aaa aba aca  bbb bcb bdb |
| 1 1 | aaa |
| 1 3 | aaa aba aca |

## \* Longest Non-Decreasing Subsequence

Write a program that reads a sequence of integers and finds in it the **longest non-decreasing subsequence**. In other words, you should remove a minimal number of numbers from the starting sequence, so that the resulting sequence is non-decreasing. In case of several longest non-decreasing sequences, print the leftmost of them. The input and output should consist of a single line, holding integer numbers separated by a space. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 | 1 |
| 7 3 5 8 -1 6 7 | 3 5 6 7 |
| 1 1 1 2 2 2 | 1 1 1 |
| 1 1 1 3 3 3 2 2 2 2 | 2 2 2 2 |
| 11 12 13 3 14 4 15 5 6 7 8 7 16 9 8 | 3 4 5 6 7 8 9 |

## Remove Names

Write a program that takes as input two lists of names and **removes from the first list all names given in the second list**. The input and output lists are given as words, separated by a space, each list at a separate line. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| Peter Alex Maria Todor Steve Diana Steve  Todor Steve Nakov | Peter Alex Maria Diana |
| Hristo Hristo Nakov Nakov Petya  Nakov Vanessa Maria | Hristo Hristo Petya |

## Join Lists

Write a program that takes as input two lists of integers and **joins them**. The result should hold all numbers from the first list, and all numbers from the second list, **without repeating numbers**, and arranged in **increasing order**. The input and output lists are given as integers, separated by a space, each list at a separate line. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 20 40 10 10 30 80  25 20 40 30 10 | 10 20 25 30 40 80 |
| 5 4 3 2 1  6 3 2 | 1 2 3 4 5 6 |
| 1  1 | 1 |

## Count of Letters

Write a program that reads a list of letters and **prints for each letter how many times it appears in the list**. The letters should be listed in alphabetical order. Use the input and output format from the examples below. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| b b a a b | a -> 2  b -> 3 |
| h d h a a a s d f d a d j d s h a a | a -> 6  d -> 5  f -> 1  h -> 3  j -> 1  s -> 2 |

## Count of Names

Write a program that reads a list of names and **prints for each name how many times it appears in the list**. The names should be listed in alphabetical order. Use the input and output format from the examples below. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| Peter Steve Nakov Steve Alex Nakov | Alex -> 1  Nakov -> 2  Peter -> 1  Steve -> 2 |
| Nakov Nakov Nakov SoftUni Nakov | SoftUni -> 1  Nakov -> 5 |

## Average Load Time Calculator

We have a report that holds dates, web site URLs and load times (in seconds) in the same format like in the examples below. Your tasks is to calculate the **average load time** for each URL. Print the URLs in the same order as they first appear in the input report. Print the output in the format given below. Use double floating-point precision. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2014-Mar-02 11:33 <http://softuni.bg> 8.37725  2014-Mar-02 11:34 <http://www.google.com> 1.335  2014-Mar-03 21:03 <http://softuni.bg> 7.25  2014-Mar-03 22:00 <http://www.google.com> 2.44  2014-Mar-03 22:01 <http://www.google.com> 2.45  2014-Mar-03 22:01 <http://www.google.com> 2.77 | <http://softuni.bg> -> 7.813625  <http://www.google.com> -> 2.24875 |
| 2014-Apr-01 02:01 <http://softuni.bg> 8.37725  2014-Apr-01 02:05 <http://www.nakov.com> 11.622  2014-Apr-01 02:06 <http://softuni.bg> 4.33  2014-Apr-01 02:11 <http://www.google.com> 1.94  2014-Apr-01 02:11 <http://www.google.com> 2.011  2014-Apr-01 02:12 <http://www.google.com> 4.882  2014-Apr-01 02:34 <http://softuni.bg> 4.885  2014-Apr-01 02:36 <http://www.nakov.com> 10.74  2014-Apr-01 02:36 <http://www.nakov.com> 11.75  2014-Apr-01 02:38 <http://softuni.bg> 3.886  2014-Apr-01 02:44 <http://www.google.com> 1.04  2014-Apr-01 02:48 <http://www.google.com> 1.4555  2014-Apr-01 02:55 <http://www.google.com> 1.977 | <http://softuni.bg> -> 5.3695625  <http://www.nakov.com> -> 11.3706666666667  <http://www.google.com> -> 2.21758333333333 |

## Longest Word in a Text

Write a program to find the longest word in a text. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| Welcome to the Software University. | University |
| The C# Basics course is awesome start in programming with C# and Visual Studio. | programming |

## Extract URLs from Text

Write a program that extracts and prints all URLs from given text. URL can be in only two formats:

* [**http://something**](http://something), e.g. <http://softuni.bg>, <http://forums.softuni.bg>, <http://www.nakov.com>
* [**www.something.domain**](http://www.something.domain), e.g. [www.nakov.com](http://www.nakov.com), [www.softuni.bg](http://www.softuni.bg), [www.google.com](http://www.google.com)

Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| The site nakov.com can be access from <http://nakov.com> or [www.nakov.com](http://www.nakov.com). It has subdomains like mail.nakov.com and svetlin.nakov.com. Please check <http://blog.nakov.com> for more information. | <http://nakov.com>  [www.nakov.com](http://www.nakov.com)  <http://blog.nakov.com> |

## \* Counting a Word in a Text

Write a program that counts **how many times a given word occurs in given text**. The first line in the input holds the word. The second line of the input holds the text. The output should be a single integer number – the number of word occurrences. Matching should be **case-insensitive**. Note that **not all matching substrings are words** and should be counted. A **word** is a sequence of letters separated by punctuation or start / end of text. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| hi  **Hi**dden networks say “**Hi**” only to **Hi**tachi devices. **Hi**, said Matu**hi**. **HI**! | 3 |
| SoftUni  The Software University (**SoftUni**) trains software engineers, gives them a profession and a job. Visit us at [http://**softuni**.bg](http://softuni.bg). Enjoy the SoftUnification at **SoftUni**.BG. Contact us.Email: [INFO@**SOFTUNI**.BG](mailto:INFO@SOFTUNI.BG). Facebook: <https://www.facebook.com/SoftwareUniversity>. YouTube: <http://www.youtube.com/SoftwareUniversity>. Google+: <https://plus.google.com/+SoftuniBg/>. Twitter: <https://twitter.com/softunibg>. GitHub: [https://github.com/**softuni**](https://github.com/softuni) | 5 |
| SoftUni  Software University | 0 |
| SoftUni  **SoftUni** | 1 |
| SoftUni  **SoftUni**.**SoftUni** | 2 |

## \* Perimeter and Area of Polygon

Write a program that **calculates the perimeter and the area of given polygon** (not necessarily convex) consisting of n floating-point coordinates in the 2D plane. Print the result rounded to two decimal digits after the decimal point. Use the input and output format from the examples. To hold the points, define a class Point(x, y). To hold the polygon use a Polygon class which holds a **list of points**. Find in Internet how to calculate the [polygon perimeter](http://www.mathopenref.com/polygonperimeter.html) and how to calculate the [area of a polygon](http://www.mathopenref.com/coordpolygonarea.html). Examples:

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 3  0 0  0 1  1 1 | perimeter = 3.41  area = 0.50 |  |
| 7  -2 1  1 3  5 1  1 2  1 1  3 -2  -2 1 | perimeter = 22.64  area = 9.5 |  |

# Exam problems.\*\*

**All of the problems below are given from Variant 8 of C# Basics Practical Exam (14 April 2014 Evening). You can submit your solutions** [**HERE**](http://judge.softuni.bg/Contests/7/CSharp-Basics-Exam-12-April-2014-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://softuni.bg/downloads/svn/csharp-basics/Feb-2014/9.%20CSharp-Basics-Exam-April-2014-Variant-8.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.

## \*\* – Inside the Building

In Absurdistan the **buildings** look like the figure on the right. They consist of **6 blocks of size h \* h**. Their bottom-left corner is located at the coordinates (0, 0). See the figure (for h = 2) to get a better idea.

Write a program that enters a size **h** and 5 points {**x1**, **y1**},{**x2**, **y2**},{**x3**, **y3**},{**x4**, **y4**}, and {**x5**, **y5**} and prints for each of the points whether it is inside or outside of the building. Points at the building's border, like {0, 0}, are considered inside.

### Input

The input data should be read from the console.

* At the first line an integer number **h** specifying the **size** of the building will be given.
* At the next 10 lines the numbers **x1**, **y1**, **x2**, **y2**, **x3**, **y3**, **x4**, **y4**, **x5**, **y5** are given.

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 5 lines. At each line print either "**inside**" or "**outside**" depending of where each of the 5 input points are located.

### Constraints

* All numbers in the input will be integers in the range [-1000 … 1000].
* Allowed working time for your program: 0.1 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** | **Comments** |  | **Input** | **Output** | **Comments** |
| 2  3  10  0  6  2  2  3  1  6  0 | outside  outside  inside  inside  inside |  | 15  29  38  37  19  30  0  -4  7  13  57 | inside  outside  inside  outside  outside |  |

## \*\* – Student Cables

Once at the Software University (SoftUni) we had problems with the Wi-Fi network. It was working well in the previous days even with a few hundred students browsing Internet in the same time, but at the exam all the students came with their laptop at fixed time and tried to establish a wireless connection simultaneously. This flooded the Wi-Fi access points and as a result some of the students were unable to get an IP address from the DHCP server. They of course established a Wi-Fi Internet connection after a few reconnects in next 5-10 minutes, but were highly stressed because they didn't had Internet immediately before the start the exam start.

Nakov, the main driver of SoftUni, decided to solve the problem by connecting some of the students through a standard **network cables**. He installed a few network switches in the exam lab and started to prepare cables for the students. His idea was to use **5 meters long cables** (called **student cables**) between the switches and the student's laptops. Nakov wanted to create as much as possible cablesof size 5 meters. He had a lot of cables of different sizes, e.g. a big roll of 300 meters, another big roll of 130 meters and a few small cables of 30 cm, 15 cm and 10 cm. The cables had **different sizes** and was measured in **different** **measures** (meters or centimeters). Nakov calculated that he needed **2 cm for crimping each RJ45 connector** and **3 cm for joining each two pieces of cable**. It was complex to calculate how much cables Nakov can create so he needs your help.

Write a program that takes as an input a **sequence of N cables of different sizes** and calculates **how many student cables** Nakov can create by first joining them all together, then cut them into 5 meters and 4 cm, and finally crimp the RJ45 connectors to obtain 5 meters long student network cables. Calculate also **the length of the unused remaining cable**. Note that cables **shorter than 20 cm** in the input will be thrown away, so please discard therm.

### Input

The input data should be read from the console.

* At the **first line** an integer number **n** specifying the **number of cables** will be given.
* At the next **2 \* n lines** the cables will be given: first comes the **cable length**; second comes the **measure**.

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 2 lines:

* The first line should hold the **number of student cables**.
* The second line should hold the **length of the remaining cable**.

### Constraints

* The number **n** will be integer in the range [1 … 100].
* The **cable length** is integer in the range [1 … 500].
* The **cable measure** is one of the following values: **meters**, **centimeters**.
* Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Output** | **Comments** |  | **Input** | **Output** |
| 4  11  meters  18  centimeters  8  meters  120  centimeters | 3  502 | We have **4** cables: **1100** cm, **18** cm, **800** cm and **120** cm. The 18 cm cable is too short (18 cm < 20 cm), so it is discarded. We join 1100 cm + 800 cm + 120 cm and we lose 2\*3 = 6 cm. We obtain **2014** cm joined cable. We create **3 student cables**: 3 \* (5 m cable + 2 cm RJ crimp + 2 cm RJ crimp) = 3 \* 504 = **1512** cm. The remainder is 2014 – 1512 = **502 cm**. | 3  116  centimeters  4  meters  20  centimeters | 1  26 |

A

BCD

EFGAB

CDEFGAB

CDEFG

ABC

D

E

FGA

BCDEF

GABCDEF

GABCD

EFG

A

## \*\* – Programmer DNA

The secret scientists from the institute of Bizarre Artificial Neurobiology or B.A.N have tried for years to find the DNA markers of the perfect programmer. The work is going well but after an incident with the printer in the institute it is impossible to print the latest discovery. That is why they have asked you to help them.

Your task is to make a program that can **print simple DNA chains of various lengths**. Simple DNA chains consist of **sequence of diamond blocks containing only letters from A to G** (see the example on the right).

Letters are chained alphabetically: A is followed by B, then C, D, E, F, G, then again A and so on. Each DNA block is with size 7.

### Input

The input data should be read from the console.

* On the first line an integer number **N** specifying the **length** of the DNA chain will be given.
* On the second line the **starting letter** of the chain will be given (capital letter from **A** to **G**).

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. Following the examples below print exactly **N** lines of the programmer's DNA. Use only capital letters from **A** to **G** and “**.**” for the empty space.

### Constraints

* **N** will always be a positive number between **7** and **999** inclusive.
* Allowed working time for your program: 0.1 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 7  B | ...B...  ..CDE..  .FGABC.  DEFGABC  .DEFGA.  ..BCD..  ...E... | 10  F | ...F...  ..GAB..  .CDEFG.  ABCDEFG  .ABCDE.  ..FGA..  ...B...  ...C...  ..DEF..  .GABCD. |

## \*\* – Magic Car Numbers

Cars in Sofia have registration numbers in format "**CA*abcdXY***" where ***a***, ***b***, ***c*** and ***d*** are digits from 0 to 9 and ***X*** and ***Y*** are letters from the following subset of the Latin alphabet: 'A', 'B', 'C', 'E', 'H', 'K', 'M', 'P', 'T' and 'X'. Examples of car numbers from Sofia are "CA8517TX", "CA2277PC", "CA0710XC", "CA1111AC", while "CC7512FJ" in not valid car number from Sofia. Local people are keen to choose special numbers for their cars, know as **magic car numbers**. They calculate the **weight of a car number** as follows: they sum its digits and letters, assuming that letters have the following values: 'A' 🡪 10, 'B' 🡪 20, 'C' 🡪 30, 'E' 🡪 50, 'H' 🡪 80, 'K' 🡪 110, 'M' 🡪 130, 'P' 🡪 160, 'T' 🡪 200, 'X' 🡪 240. For example the weight("CA6511BM") = 30 + 10 + 6 + 5 + 1 + 1 + 20 + 130 = 203. A **magic car number** is a car number that has a **special magic weight** (e.g. 256 or 512 for developers) and its number is in some of the formats "CA*aaaaXY*", "CA*abbbXY*", "CA*aaabXY*", "CA*aabbXY*", "CA*ababXY*" and "CA*abbaXY*", where ***a*** and ***b*** are two different digits and ***X*** and ***Y*** are letters from the Latin alphabet subset { 'A', 'B', 'C', 'E', 'H', 'K', 'M', 'P', 'T', 'X' }.

Your task is to write a program that calculates **how many cars** ca n be registered in Sofia **for given magic weight**.

### Input

The input data should be read from the console. It will consist of a single value: the **magic weight**.

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 is a single value: the number of cars matching given magic value.

### Constraints

* All input numbers will be **integers** in the range [1…1000].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Matching Car Numbers** |
| 555 | 2 | CA8999XX, CA9998XX |
| 512 | 18 | CA5999TX, CA5999XT, CA7799TX, CA7979TX, CA7997TX, CA7799XT, CA7979XT, CA7997XT, CA8888TX, CA8888XT, CA9995TX, CA9977TX, CA9797TX, CA9779TX, CA9995XT, CA9977XT, CA9797XT, CA9779XT |
| 95 | 46 | CA0555AC, CA0555BB, CA0005BC, CA0555CA, CA0005CB, CA1888AB, CA1888BA, CA1112BC, CA1112CB, CA2229AC, CA2229BB, CA2111BC, CA2229CA, CA2111CB, CA3444AC, CA3336AC, CA3444BB, CA3336BB, CA3444CA, CA3336CA, CA4777AB, CA4443AC, CA4777BA, CA4443BB, CA4443CA, CA5550AC, CA5550BB, CA5000BC, CA5550CA, CA5000CB, CA6667AB, CA6333AC, CA6667BA, CA6333BB, CA6333CA, CA7774AB, CA7666AB, CA7774BA, CA7666BA, CA8999AA, CA8881AB, CA8881BA, CA9998AA, CA9222AC, CA9222BB, CA9222CA |

## \*\* – Bit Flipper

We are given a **bit sequence** in the form of **64-bit integer**. We pass through the bits from left to right and we **flip all sequences of 3 equal bits** (111 🡪 000, 000 🡪 111). For example, 8773276988229695713 represents the bit sequence 0111100111000000111100001111000000011111100010100011100011100001. We flip from left to right all 3 consecutive equal bits: 0**111**100**111000000111**1**000**0**111**1**000000**0**111111000**101**000111000111000**01 🡪 0**000**100**000111111000**1**111**0**000**1**111111**0**000000111**101**111000111000111**01. The obtained 64-bit number after flipping is 594226797558351645.

Your task is to write a program that enters a 64-bit integer, performs the above described flipping, and prints the obtained result as a 64-bit integer.

### Input

The input data should be read from the console. It consists of a single 64-bit integer number.

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

### Output

Print at the console the 64-bit integer, representing the obtained bits after the flipping.

### Constraints

* The **input number** will be a 64-bit integer in the range [0 … 18 446 744 073 709 551 615].
* Allowed working time for your program: 0.1 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 8773276988229695713 | 594226797558351645 |
| **Explanation** | |
| 8773276988229695713 🡪  0**111**100**111000000111**1**000**0**111**1**000000**0**111111000**101**000111000111000**01 🡪 0**000**100**000111111000**1**111**0**000**1**111111**0**000000111**101**111000111000111**01 🡪  594226797558351645 | |

|  |  |
| --- | --- |
| **Input** | **Output** |
| 594226797558350599 | 17222015390969265120 |
| **Explanation** | |
| 594226797558350599 🡪  **000**01**000**00**111111000111**1**000**0**111111**1**000000**0**111**10**111**1**000**011**000**00**111** 🡪 **111**01**111**00**000000111000**1**111**0**000000**1**111111**0**000**10**000**1**111**011**111**00**000** 🡪  17222015390969265120 | |

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

**C#2**

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

# Homework: Arrays

### Problem 1. Allocate array

* Write a program that allocates array of 20 integers and initializes each element by its index multiplied by 5.
* Print the obtained array on the console.

### Problem 2. Compare arrays

* Write a program that reads two integer arrays from the console and compares them element by element.

### Problem 3. Compare char arrays

* Write a program that compares two char arrays lexicographically (letter by letter).

### Problem 4. Maximal sequence

* Write a program that finds the **maximal sequence** of equal elements in an array.

Example:

| **input** | **result** |
| --- | --- |
| 2, 1, 1, 2, 3, 3, **2, 2, 2**, 1 | 2, 2, 2 |

### Problem 5. Maximal increasing sequence

* Write a program that finds the maximal increasing sequence in an array.

Example:

| **input** | **result** |
| --- | --- |
| 3, **2, 3, 4**, 2, 2, 4 | 2, 3, 4 |

### Problem 6. Maximal K sum

* Write a program that reads two integer numbers N and K and an array of N elements from the console.
* Find in the array those K elements that have maximal sum.

### Problem 7. Selection sort

* **Sorting** an array means to arrange its elements in increasing order. Write a program to sort an array.
* Use the [Selection sort](http://en.wikipedia.org/wiki/Selection_sort) algorithm: Find the smallest element, move it at the first position, find the smallest from the rest, move it at the second position, etc.

### Problem 8. Maximal sum

* Write a program that finds the sequence of maximal sum in given array.

Example:

| **input** | **result** |
| --- | --- |
| 2, 3, -6, -1, **2, -1, 6, 4**, -8, 8 | 2, -1, 6, 4 |

* Can you do it with only one loop (with single scan through the elements of the array)?

### Problem 9. Frequent number

* Write a program that finds the most frequent number in an array.

Example:

| **input** | **result** |
| --- | --- |
| **4**, 1, 1, **4**, 2, 3, **4**, **4**, 1, 2, **4**, 9, 3 | 4 (5 times) |

### Problem 10. Find sum in array

* Write a program that finds in given array of integers a sequence of given sum S (if present).

Example:

| **array** | **S** | **result** |
| --- | --- | --- |
| 4, 3, 1, **4, 2, 5**, 8 | 11 | 4, 2, 5 |

### Problem 11. Binary search

* Write a program that finds the index of given element in a sorted array of integers by using the [Binary search](http://en.wikipedia.org/wiki/Binary_search_algorithm) algorithm.

### Problem 12. Index of letters

* Write a program that creates an array containing all letters from the alphabet (A-Z).
* Read a word from the console and print the index of each of its letters in the array.

### Problem 13.\* Merge sort

* Write a program that sorts an array of integers using the [Merge sort](http://en.wikipedia.org/wiki/Merge_sort) algorithm.

### Problem 14. Quick sort

* Write a program that sorts an array of integers using the [Quick sort](http://en.wikipedia.org/wiki/Quicksort) algorithm.

### Problem 15. Prime numbers

* Write a program that finds all prime numbers in the range [1...10 000 000]. Use the [Sieve of Eratosthenes](http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes) algorithm.

### Problem 16.\* Subset with sum S

* We are given an array of integers and a number S.
* Write a program to find if there exists a subset of the elements of the array that has a sum S.

Example:

| **input array** | **S** | **result** |
| --- | --- | --- |
| 2, **1**, **2**, 4, 3, **5**, 2, **6** | 14 | yes |

### Problem 17.\* Subset K with sum S

* Write a program that reads three integer numbers N, K and S and an array of N elements from the console.
* Find in the array a subset of K elements that have sum S or indicate about its absence.

### Problem 18.\* Remove elements from array

* Write a program that reads an array of integers and removes from it a minimal number of elements in such a way that the remaining array is sorted in increasing order.
* Print the remaining sorted array.

Example:

| **input** | **result** |
| --- | --- |
| 6, **1**, 4, **3**, 0, **3**, 6, **4**, **5** | 1, 3, 3, 4, 5 |

### Problem 19.\* Permutations of set

* Write a program that reads a number N and generates and prints all the permutations of the numbers [1 … N].

Example:

| **N** | **result** |
| --- | --- |
| 3 | {1, 2, 3}  {1, 3, 2}  {2, 1, 3}  {2, 3, 1}  {3, 1, 2}  {3, 2, 1} |

### Problem 20.\* Variations of set

* Write a program that reads two numbers N and K and generates all the variations of K elements from the set [1..N].

Example:

| **N** | **K** | **result** |
| --- | --- | --- |
| 3 | 2 | {1, 1}  {1, 2}  {1, 3}  {2, 1}  {2, 2}  {2, 3}  {3, 1}  {3, 2}  {3, 3} |

### Problem 21.\* Combinations of set

* Write a program that reads two numbers N and K and generates all the combinations of K distinct elements from the set [1..N].

Example:

| **N** | **K** | **result** |
| --- | --- | --- |
| 5 | 2 | {1, 2}  {1, 3}  {1, 4}  {1, 5}  {2, 3}  {2, 4}  {2, 5}  {3, 4}  {3, 5}  {4, 5} |

# Homework: Multidimensional Arrays

### Problem 1. Fill the matrix

* Write a program that fills and prints a matrix of size (n, n) as shown below:

Example for *n=4*:

| **a)** | **b)** | **c)** | **d)\*** |
| --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | 1 | 5 | 9 | 13 | | 2 | 6 | 10 | 14 | | 3 | 7 | 11 | 15 | | 4 | 8 | 12 | 16 | | |  |  |  |  | | --- | --- | --- | --- | | 1 | 8 | 9 | 16 | | 2 | 7 | 10 | 15 | | 3 | 6 | 11 | 14 | | 4 | 5 | 12 | 13 | | |  |  |  |  | | --- | --- | --- | --- | | 7 | 11 | 14 | 16 | | 4 | 8 | 12 | 15 | | 2 | 5 | 9 | 13 | | 1 | 3 | 6 | 10 | | |  |  |  |  | | --- | --- | --- | --- | | 1 | 12 | 11 | 10 | | 2 | 13 | 16 | 9 | | 3 | 14 | 15 | 8 | | 4 | 5 | 6 | 7 | |

### Problem 2. Maximal sum

* Write a program that reads a rectangular matrix of size N x M and finds in it the square 3 x 3 that has maximal sum of its elements.

### Problem 3. Sequence n matrix

* We are given a matrix of strings of size N x M. Sequences in the matrix we define as sets of several neighbour elements located on the same line, column or diagonal.
* Write a program that finds the longest sequence of equal strings in the matrix.

Example:

| **matrix** | **result** |  | **matrix** | **result** |
| --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **ha** | fifi | ho | hi | | fo | **ha** | hi | xx | | xxx | ho | **ha** | xx | | ha, ha, ha |  | |  |  |  | | --- | --- | --- | | s | qq | **s** | | pp | pp | **s** | | pp | qq | **s** | | s, s, s |

### Problem 4. Binary search

* Write a program, that reads from the console an array of N integers and an integer K, sorts the array and using the method Array.BinSearch() finds the largest number in the array which is ≤ K.

### Problem 5. Sort by string length

* You are given an array of strings. Write a method that sorts the array by the length of its elements (the number of characters composing them).

### Problem 6.\* Matrix class

* Write a class Matrix, to hold a matrix of integers. Overload the operators for adding, subtracting and multiplying of matrices, indexer for accessing the matrix content and ToString().

### Problem 7.\* Largest area in matrix

* Write a program that finds the largest area of equal neighbour elements in a rectangular matrix and prints its size.

Example:

| **matrix** | **result** |
| --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 1 | **3** | 2 | 2 | 2 | 4 | | **3** | **3** | **3** | 2 | 4 | 4 | | 4 | **3** | 1 | 2 | **3** | **3** | | 4 | **3** | 1 | **3** | **3** | 1 | | 4 | **3** | **3** | **3** | 1 | 1 | | 13 |

Hint: you can use the algorithm [*Depth-first search*](http://en.wikipedia.org/wiki/Depth-first_search) or [*Breadth-first search*](http://en.wikipedia.org/wiki/Breadth-first_search).

# Homework: Methods

### Problem 1. Say Hello

* Write a method that asks the user for his name and prints “Hello, <name>”
* Write a program to test this method.

Example:

| **input** | **output** |
| --- | --- |
| Peter | Hello, Peter! |

### Problem 2. Get largest number

* Write a method GetMax() with two parameters that returns the larger of two integers.
* Write a program that reads 3 integers from the console and prints the largest of them using the method GetMax().

### Problem 3. English digit

* Write a method that returns the last digit of given integer as an English word.

Examples:

| **input** | **output** |
| --- | --- |
| 512 | two |
| 1024 | four |
| 12309 | nine |

### Problem 4. Appearance count

* Write a method that counts how many times given number appears in given array.
* Write a test program to check if the method is workings correctly.

### Problem 5. Larger than neighbours

* Write a method that checks if the element at given position in given array of integers is larger than its two neighbours (when such exist).

### Problem 6. First larger than neighbours

* Write a method that returns the index of the first element in array that is larger than its neighbours, or -1, if there’s no such element.
* Use the method from the previous exercise.

### Problem 7. Reverse number

* Write a method that reverses the digits of given decimal number.

Example:

| **input** | **output** |
| --- | --- |
| 256 | 652 |
| 123.45 | 54.321 |

### Problem 8. Number as array

* Write a method that adds two positive integer numbers represented as arrays of digits (each array element arr[i] contains a digit; the last digit is kept in arr[0]).
* Each of the numbers that will be added could have up to 10 000 digits.

### Problem 9. Sorting array

* Write a method that return the maximal element in a portion of array of integers starting at given index.
* Using it write another method that sorts an array in ascending / descending order.

### Problem 10. N Factorial

* Write a program to calculate n! for each n in the range [1..100].

Hint: Implement first a method that multiplies a number represented as array of digits by given integer number.

### Problem 11. Adding polynomials

* Write a method that adds two polynomials.
* Represent them as arrays of their coefficients.

Example:

x2 + 5 = 1x2 + 0x + 5 => {5, 0, 1}

### Problem 12. Subtracting polynomials

* Extend the previous program to support also subtraction and multiplication of polynomials.

### Problem 13. Solve tasks

* Write a program that can solve these tasks:
  + Reverses the digits of a number
  + Calculates the average of a sequence of integers
  + Solves a linear equation a \* x + b = 0
* Create appropriate methods.
* Provide a simple text-based menu for the user to choose which task to solve.
* Validate the input data:
  + The decimal number should be non-negative
  + The sequence should not be empty
  + a should not be equal to 0

### Problem 14. Integer calculations

* Write methods to calculate minimum, maximum, average, sum and product of given set of integer numbers.
* Use variable number of arguments.

### Problem 15.\* Number calculations

* Modify your last program and try to make it work for any number type, not just integer (e.g. decimal, float, byte, etc.)
* Use generic method (read in Internet about generic methods in C#).

# Homework: Numeral Systems

### Problem 1. Decimal to binary

* Write a program to convert decimal numbers to their binary representation.

### Problem 2. Binary to decimal

* Write a program to convert binary numbers to their decimal representation.

### Problem 3. Decimal to hexadecimal

* Write a program to convert decimal numbers to their hexadecimal representation.

### Problem 4. Hexadecimal to decimal

* Write a program to convert hexadecimal numbers to their decimal representation.

### Problem 5. Hexadecimal to binary

* Write a program to convert hexadecimal numbers to binary numbers (directly).

### Problem 6. Binary to hexadecimal

* Write a program to convert binary numbers to hexadecimal numbers (directly).

### Problem 7. One system to any other

* Write a program to convert from any numeral system of given base s to any other numeral system of base d (2 ≤ s, d ≤ 16).

### Problem 8. Binary short

* Write a program that shows the binary representation of given 16-bit signed integer number (the C# type short).

### Problem 9.\* Binary floating-point

* Write a program that shows the internal binary representation of given 32-bit signed floating-point number in IEEE 754 format (the C# type float).

Example:

| **number** | **sign** | **exponent** | **mantissa** |
| --- | --- | --- | --- |
| -27,25 | 1 | 10000011 | 10110100000000000000000 |