# 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 2 2 2 |
| 1 1 1 3 3 3 2 2 2 2 | 1 1 1 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 -> 4 |

## 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 | |