**Такмичење из програмирања 2022-2023 – 1. круг квалификација**

**Задатак 01 – А – Судар бројева**

**[https://arena.petlja.org/sr-Latn-RS/competition/2022-2023-kv1-testiranje#tab\_134872](https://arena.petlja.org/sr-Latn-RS/competition/2022-2023-kv1-testiranje%23tab_134872)**

[**https://algora.petlja.org/t/2022-2023/7670**](https://algora.petlja.org/t/2022-2023/7670)

| **Vremensko ograničenje** | **Memorijsko ograničenje** | **ulaz** | **izlaz** |
| --- | --- | --- | --- |
| 0,1 s | 32 MB | standardni ulaz | standardni izlaz |

Мала Даница, потпуно револтирана одлуком Комисије да ове године укине један круг квалификација, кренула је да разбија све око себе, укључујући и бројеве који јој падну под руку. Наиме, када угледа два природна броја она баци један број на други тако да се бројеви сударе. Сударом два броја, настаје нов природан број.

Приликом судара два природна броја X и Y, свака цифра једног броја упоређује се с одговарајућом цифром другог броја, то јест, јединица се упореди с јединицом, десетица с десетицом, стотина са стотином, итд. Мања од две цифре у том поређењу испадне, док већа улази у састав новоформираног броја. Уколико су одговарајуће цифре једнаке, приликом судара обе улазе у новостворени број те ће стога он имати више цифара од X и Y. Уколико у неком од два природна броја која се сударају нема одговарајуће цифре, сматра се да та цифра не постоји те се без поређења увек узима цифра другог броја, па макар она била и нула.

Пошто је бројева превише, мала Даница не може сама да се избори с њима, па вас је замолила да јој помогнете тако што ћете сударати бројеве заједно с њом. Да бисте то учинили морате одговорити на питање који ће се број створити приликом судара два природна броја?.

**Опис улаза**

У првом реду стандардног улаза се налазе први природан број X, док се у другом реду стандардног улаза налази други природан број Y. Ова два природна броја треба сударити на начин описан у тексту задатка.

**Опис излаза**

У једином реду стандардног излаза исписати новостворени број који се добија приликом судара два броја са улаза.

**Пример 1**

**Улаз**

73

28

**Излаз**

78

**Пример 2**

**Улаз**

64

357

**Излаз**

367

**Пример 3**

**Улаз**

234

135

**Излаз**

2335

**Пример 4**

**Улаз**

99099

9999

**Излаз**

99999999

**Пример 5**

**Улаз**

2

100

**Излаз**

102

**Објашњења примера**

У првом примеру се сударају бројеви 73 и 28. Оба броја су двоцифрена, те се међусобно упоређују цифре десетица, односно 7 и 2, и цифре јединица, односно 3 и 8, и од већих цифара ствара се нови број који има вредност 78, пошто је 7 веће од 2, а 8 веће од 3.

У другом примеру сударају се бројеви 64 и 357. Први број је двоцифрен, а други троцифрен, па се стога цифра стотина, 3, преузима из већег броја. Затим се приликом судара међусобно упоређују цифре десетица, 6 и 5, као и цифре јединица 4 и 7, и од већих цифара формира се нови број који има вредност 367, пошто је 3 преузето, а 6 веће од 5 и 7 веће од 4.

У трећем примеру сударају се троцифрени бројеви 234 и 135. Међусобно се упоређују цифре стотина, 2 и 1, цифре десетица 3 и 3, и коначно цифре јединица 4 и 5. Поред одговарајућих већих цифара које сачињавају нов број, а то су цифре 2 и 5, у састав новог броја улазе и све међусобно једнаке цифре на одговарајућим местима, односно у овом конкретном случају обе цифре 3 с места десетица, па је вредност новоствореног броја 2335.

У четвртом примеру сударају се бројеви 99099 и 9999. Како је први петоцифрени, а други четвороцифрени, пореде се четири најниже цифре, а по највиша се цифра преузима из првог броја. Приликом упоређивања постоје само два случаја: у првом се пореди 9 и 0 те се у новоформираном броју преноси једна деветка, а у другом случају се пореди 9 и 9 па у новостворени број улазе обе деветке. Стога је резултат судара 99999999 јер у два почетна броја имамо укупно осам деветки и све оне улазе у састав резултујућег броја.

Најзад, у петом тест примеру сударају се бројеви 2 и 100. Пошто је први број једноцифрен, а други троцифрен, то се прве две цифре 1 и 0 свакако преузимају из другог броја, а у поређењу цифара јединица, 2 је веће од 0, те се коначно добија вредност 102.

**Ограничења**

* 1 ≤ X, Y ≤ 109

Тест примери су подељени у пет дисјунктних група:

* У тест примерима вредним 10 поена: 1 ≤ X, Y ≤ 10, односно X и Y су једноцифрени.
* У тест примерима вредним 15 поена: 10 ≤ X, Y ≤ 100, односно X и Y су двоцифрени.
* У тест примерима вредним 15 поена: 100 ≤ X, Y ≤ 1000, односно X и Y су трооцифрени.
* У тест примерима вредним 20 поена: 103 ≤ X, Y ≤ 109 и да X и Y имају једнак број цифара.
* У тест примерима вредним 40 поена нема додатних ограничења.

**Напомена**

Задати и исписани бројеви су увек у декадном запису без водећих нула.

1. Такмичење из програмирања 2022/2023 - 1. круг квалификација - 1. задатак А – Судар бројева: <https://arena.petlja.org/sr-Latn-RS/competition/2022-2023-kv1-testiranje#tab_134872>   
   <https://algora.petlja.org/t/2022-2023/7670>

**// C# - Решење за 069 поена**

// Tkm 2022\_23 01 KV 01 Zad 01 A Sudar brojeva

// https://takprog.petlja.org/srednjaskola/post/2022-11-21-kv1-preliminarni-rezultati

// https://arena.petlja.org/sr-Latn-RS/competition/2022-2023-kv1-testiranje#tab\_134872

using System;

class Tkm\_2022\_23\_K01\_Z01\_A\_Sudar\_brojeva\_Bod\_069

{

static void Main()

{

int x = int.Parse(Console.ReadLine()); // 1 <= x <= 10^9 = 1000000000

int y = int.Parse(Console.ReadLine()); // 1 <= y <= 10^9 = 1000000000

// x = 1000000000; // Iz uslova (ogranicenja) zadatka da je najveca moguca ulazna vrednost 10^9 = 1000000000

// y = 1000000000; // Iz uslova (ogranicenja) zadatka da je najveca moguca ulazna vrednost 10^9 = 1000000000

if (x < y) // Obezbedjumo da veca ulazna vrednost bude u X, a manja u Y

{

int p = x;

x = y;

y = p;

}

int r = 0; // Rezultat -> inicijalna vrednost (ne moze biti manja od 1, odnosno od 0

// r = 11000000000000000000;

int d = 1; // Koeficijent za mnozenje (jedinice, desetice, stotine...)

while (x > 0) // Klasican algoritam za rastavljanje celog broja na cifre

{

int cx = x % 10;

int cy = y % 10;

int cr = cx;

if (cy > cx) { cr = cy; }

r = r + cr \* d;

if (cx == cy && y > 0) // Ukoliko su obe cifre jednake onda obe dopisujemo.

{

d = d \* 10;

r = r + cr \* d;

}

d = d \* 10;

x = x / 10;

y = y / 10;

}

Console.WriteLine(r);

}

}

**Напомена:**

Приликом тестирања (извршавања) овог програмског кода примећујемо да ће за граничне вредности улазних података (како су наведене у услову/захтеву задатка) односно вредности 1000000000 за обе улазне променљиве (иако су обе максималне вредносту у опсегу целобројног типа **int**) програм исписивати погрешне, неисправне, некоректне излазне вредности (углавном негативне) уместо очекиване **11000000000000000000** (две јединице иза којих следи 18 нула, односно 11\*1018 или 1.1\*1019) .

Самим тим можемо да закључујемо да ово решење програмског кода неће проћи све тест примере, те неће имати максималан број бодова. На основу услова наведених у задатку можемо очекивати да ће овај програмски код проћи све тест примере за једноцифрене, двоцифрене и троцифрене бројеве, односно да ће имати најамање 40, али не и 100 бодова.

Да би исправно израчунали вредности бројева већих од 109 (а мањих од 1018) потребно је да уместо типа података **int** користимо тип **Int64**, као у следећој верзији програмског кода:

**// C# - Решење за 097 поена**

// Tkm 2022\_23 01 KV 01 Zad 01 A Sudar brojeva

// https://takprog.petlja.org/srednjaskola/post/2022-11-21-kv1-preliminarni-rezultati

// https://arena.petlja.org/sr-Latn-RS/competition/2022-2023-kv1-testiranje#tab\_134872

using System;

class Tkm\_2022\_23\_K01\_Z01\_A\_Sudar\_brojeva\_Bod\_097

{

static void Main()

{

int x = int.Parse(Console.ReadLine()); // 1 <= x <= 10^9 = 1000000000

int y = int.Parse(Console.ReadLine()); // 1 <= y <= 10^9 = 1000000000

// x = 1000000000; // Iz uslova (ogranicenja) zadatka da je najveca moguca ulazna vrednost 10^9 = 1000000000

// y = 1000000000; // Iz uslova (ogranicenja) zadatka da je najveca moguca ulazna vrednost 10^9 = 1000000000

if (x < y) // Obezbedjumo da veca ulazna vrednost bude u X, a manja u Y

{

int p = x;

x = y;

y = p;

}

Int64 r = 0; // Rezultat -> inicijalna vrednost (ne moze biti manja od 1, odnosno od 0

// r = 11000000000000000000;

Int64 d = 1; // Koeficijent za mnozenje (jedinice, desetice, stotine...)

while (x > 0) // Klasican algoritam za rastavljanje celog broja na cifre

{

int cx = x % 10;

int cy = y % 10;

int cr = cx;

if (cy > cx) { cr = cy; }

r = r + cr \* d;

if (cx == cy && y > 0) // Ukoliko su obe cifre jednake onda obe dopisujemo.

{

d = d \* 10;

r = r + cr \* d;

}

d = d \* 10;

x = x / 10;

y = y / 10;

}

Console.WriteLine(r);

}

}

**Напомена:**

Приликом тестирања (извршавања) овог програмског кода примећујемо да добијамо исправне (коректне) резултате, односно излазне вредности које су мање од 1018, односно које имају највише 19 цифара.

Ипак, баш за граничне вредности улазних података, односно вредности 1000000000 за обе улазне променљиве (иако су обе максималне вредносту у опсегу целобројног типа **int**) програм исписивати погрешaн, неисправaн, некоректaн резултат (и то негативан) уместо очекиваног **11000000000000000000** (две јединице иза којих следи 18 нула, односно 11\*1018 или 1.1\*1019).

Самим тим можемо да закључујемо да ово решење програмског кода неће проћи све тест примере, те неће имати максималан број бодова. На основу услова наведених у задатку можемо очекивати да ће овај програмски код проћи знатно већи број тест примера, али не и све. Заиста, програмски код пролази 44 од свих 45 тест примера, сем [**тест примера 29**](https://nyc3.digitaloceanspaces.com/takprog/2022-23/2022-2023-kv1-problems.zip).

Да би исправно израчунали вредности бројева већих од 109 (а мањих од 1018) потребно је да уместо типа података **Int64** користимо тип **UInt64** и да уместо типа података **int** користимо тип **uint**, као у следећој верзији програмског кода:

**// C# - Решење за 100 поена – UInt64**

// Tkm 2022\_23 01 KV 01 Zad 01 A Sudar brojeva

// https://takprog.petlja.org/srednjaskola/post/2022-11-21-kv1-preliminarni-rezultati

// https://arena.petlja.org/sr-Latn-RS/competition/2022-2023-kv1-testiranje#tab\_134872

using System;

class Tkm\_2022\_23\_K01\_Z01\_A\_Sudar\_brojeva\_Bod\_100

{

static void Main()

{

uint x = uint.Parse(Console.ReadLine()); // 1 <= x <= 10^9 = 1000000000

uint y = uint.Parse(Console.ReadLine()); // 1 <= y <= 10^9 = 1000000000

// x = 1000000000; // Iz uslova (ogranicenja) zadatka da je najveca moguca ulazna vrednost 10^9 = 1000000000

// y = 1000000000; // Iz uslova (ogranicenja) zadatka da je najveca moguca ulazna vrednost 10^9 = 1000000000

if (x < y) // Obezbedjumo da veca ulazna vrednost bude u X, a manja u Y

{

uint p = x;

x = y;

y = p;

}

UInt64 r = 0; // Rezultat -> inicijalna vrednost (ne moze biti manja od 1, odnosno od 0

// r = 11000000000000000000;

UInt64 d = 1; // Koeficijent za mnozenje (jedinice, desetice, stotine...)

while (x > 0) // Klasican algoritam za rastavljanje celog broja na cifre

{

uint cx = x % 10;

uint cy = y % 10;

uint cr = cx;

if (cy > cx) { cr = cy; }

r = r + cr \* d;

if (cx == cy && y > 0) // Ukoliko su obe cifre jednake onda obe dopisujemo.

{

d = d \* 10;

r = r + cr \* d;

}

d = d \* 10;

x = x / 10;

y = y / 10;

}

Console.WriteLine(r);

}

}

**Напомена:**

Сада програмски код пролази свих 45 тест примера за максималних 100 поена.

Када имамо велике целе бројеве (веће од 1018), односно целе бројеве са по неколико десетина цифара, онда је једно од решења да користимо низове цифара, стрингове или да уместо типа података **UInt64** користимо и посебан целобројни тип података **BigInteger**, као у следећој верзији кода:

**// C# - Решење за 100 поена – BigInteger**

// Tkm 2022\_23 01 KV 01 Zad 01 A Sudar brojeva

// https://takprog.petlja.org/srednjaskola/post/2022-11-21-kv1-preliminarni-rezultati

// https://arena.petlja.org/sr-Latn-RS/competition/2022-2023-kv1-testiranje#tab\_134872

using System;

using System.Numerics; // BigInteger;

class Tkm\_2022\_23\_K01\_Z01\_A\_Sudar\_brojeva\_Bod\_100

{

static void Main()

{

int x = int.Parse(Console.ReadLine()); // 1 <= x <= 10^9 = 1000000000

int y = int.Parse(Console.ReadLine()); // 1 <= y <= 10^9 = 1000000000

// x = 1000000000; // Iz uslova (ogranicenja) zadatka da je najveca moguca ulazna vrednost 10^9 = 1000000000

// y = 1000000000; // Iz uslova (ogranicenja) zadatka da je najveca moguca ulazna vrednost 10^9 = 1000000000

if (x < y) // Obezbedjumo da veca ulazna vrednost bude u X, a manja u Y

{

int p = x;

x = y;

y = p;

}

BigInteger r = 0; // Rezultat -> inicijalna vrednost (ne moze biti manja od 1, odnosno od 0

// r = 11000000000000000000;

BigInteger d = 1; // Koeficijent za mnozenje (jedinice, desetice, stotine...)

while (x > 0) // Klasican algoritam za rastavljanje celog broja na cifre

{

int cx = x % 10;

int cy = y % 10;

int cr = cx;

if (cy > cx) { cr = cy; }

r = r + cr \* d;

if (cx == cy && y > 0) // Ukoliko su obe cifre jednake onda obe dopisujemo.

{

d = d \* 10;

r = r + cr \* d;

}

d = d \* 10;

x = x / 10;

y = y / 10;

}

Console.WriteLine(r);

}

}

**Напомена:**

Када имамо велике целе бројеве (веће од 1018), односно целе бројеве са по неколико десетина цифара, онда је једно од решења да користимо низове цифара или стрингове, као у следећој верзији кода:

**// C# - Решење за 100 поена – String**

// Tkm 2022\_23 01 KV 01 Zad 01 A Sudar brojeva

// https://takprog.petlja.org/srednjaskola/post/2022-11-21-kv1-preliminarni-rezultati

// https://arena.petlja.org/sr-Latn-RS/competition/2022-2023-kv1-testiranje#tab\_134872

using System;

class Tkm\_2022\_23\_K01\_Z01\_A\_Sudar\_brojeva\_Bod\_100

{

static void Main()

{

String x = Console.ReadLine(); // 1 <= x <= 10^9 // x = "1000000000";

String y = Console.ReadLine(); // 1 <= x <= 10^9 // y = "1000000000";

if (x.Length < y.Length)

{

String p = x;

x = y;

y = p;

}

int nx = x.Length;

int ny = y.Length;

String r = ""; // r = "11000000000000000000";

for (int i = ny - 1; i >= 0; i--)

{

// Console.WriteLine(x[i + nx - ny] + " " + y[i]);

if (x[i + nx - ny] < y[i])

{

r = y[i] + r;

}

else

{

r = x[i + nx - ny] + r;

}

if (x[i + nx - ny] == y[i])

{

r = y[i] + r;

}

}

for (int i = nx - ny - 1; i >= 0; i--)

{

// Console.WriteLine(x[i]);

r = x[i] + r;

}

Console.WriteLine(r);

}

}

**Напомена:**

Када имамо велике целе бројеве (веће од 1018), односно целе бројеве са по неколико десетина цифара, онда је једно од решења да користимо низове цифара или стрингове, као у следећој верзији кода:...