**Problem 2 – Petar's Game**

Petar likes to play with numbers. He is very good with **strings** too. One day he decided to invent a new game of **summing numbers.** He will get one number and will try to divide it by 5**, if the number can be divided without remainder** (for example 15 can be divided by 5 without remainder, but 17 divided by 5 is 3 with remainder 2) Petar will **add this number to the sum.** However **if the number cannot be divided without remainder, only the remainder will be added to the sum.** After he is done with the numbers**,** Petar likes to **replace some of the sum's digits with strings**. If the sum is **odd** he will replace **the last digit and all others that are the same as it** with a given string. If the sum is **even** **he will do the same, but with the first digit.** For example, if the sum is 2434, and the string is "a" - the result will be a434.

**You will be given a start number, an end number and a string.** You have to check all numbers between the start number and the end number (without the end number), do Petar's algorithm and finally replace the digits with the string as described above.

**Input**

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

* The first line will hold the starting number;
* The second line will hold the end number;
* The third like will hold the replacement string

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

**Output**

* The output data must be printed on the console.
* On the only output line you must print the result of the game.

**Constraints**

* Start number and end number will be integers in the range [0 … 18 446 744 073 709 551 615].
* The string will contain letters and numbers.
* Allowed memory: 16 MB.

**Examples**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 10  14  a | a6 | 10 + 1 + 2 + 3 = 16. 16 is even 1 will be replaced. Result is a6 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** |
| 10  99  as | asas2as | 234  3547  SadPanda | 12SadPanda969SadPanda |

**Solution explanation**

**Input**

Firstly we need to correctly read the input from the console. We need to save the given information in the correct data type variable.

* The **start number** comes as **ulong integer** **number**. So we **declare** one **ulong variable** and we **save** the first line of input in it (ulong.Parse)
* The **end number** comes as **ulong integer** **number**. So we **declare** another **ulong variable** and we **save** the second line of input in it (ulong.Parse)
* The **replacement string** is a string. So we need to **declare** a **string** **variable** and **save** the next line of input in it.
* To save the **sum** information we need a bigger variable than ulong. So we **declare** a **decimal** **variable** and **assign** it with **0**.
* To save the **digit** that we will eventually **be** **replaced** we need to **declare** a **string** **variable** because we need to replace a string with another string.

For now the code should look like this:

|  |
| --- |
| ulong startNumber = ulong.Parse(Console.ReadLine());  ulong endNumber = ulong.Parse(Console.ReadLine());  string magicString = Console.ReadLine();  string replacedDigit;  decimal sum = 0; |

**Program logic**

Now we need to write the program logic. For this program we need to calculate the sum of some numbers.

* To calculate the **sum** we need first to **loop** through **all** the **numbers** from the **start** number to (but not including) the **end** number. E.g. we can use for-loop.
* We are going to use the **iterator** as a **control** **variable** that will **simulate** all **values** between the **start** and the **end** **number**. So we need to **assign** the **iterator** with **ulong** type as well.
* In the **body** ofthe **for-loop** we will use an **if-statement** to **check** whether the **current** **number** (the iterator) can be **divided** **by 5** **without** any **remainder**. If so, we need to **add** the **current** **number** to the **sum**. If the **number** **cannot** be **divided**, we **add** only **the** **remainder** of the **division** by 5.

For now the code should look like this:

|  |
| --- |
| for (ulong i = startNumber; i < endNumber; i++)  {  if (i % 5 == 0)  {  sum += i;  }  else  {  sum += i % 5;  }  } |

* Now we need to **check** whether the **sum** is an **odd** or **even** number. We use a **conditional** **statement** and try a **division** **by 2**. If the **division** **has** a **remainder** – the number is **odd**, if it **has** **no** **remainder** – the number is **even**.
* If the **sum** is **odd** we need to find the **last** **digit** of the sum. To do that we **convert** the **number** to a **string** and **get** the **last** **character** of it and **convert** the **character** to a **string** so that we can replace it later.
* If the **sum** is **even** we need to find the **first** **digit** of the sum. To do that we need to **convert** the **number** to a **string** and **get** the **first** **character** of it and **convert** the **character** to a **string** so that we can replace it later.

For now the code should look like this:

|  |
| --- |
| if (sum % 2 == 1)  {  replacedDigit = sum.ToString()[sum.ToString().Length - 1].ToString();  }  else  {  replacedDigit = sum.ToString()[0].ToString();  } |

**Output**

Finally we need to print out the resulting string.

* First we need to **replace** the **digits** of the **sum** with the **magic** **string**. We use a **Replace()** method that will **replace** **all** **occurring** **substrings** with a **given** **string**. For this occasion we replace the replaced digit with the replacement string.
* After that we **print** the **resulting** **string** to the **console**.

For now the code should look like this:

|  |
| --- |
| Console.WriteLine(sum.ToString().Replace(replacedDigit, magicString)); |