**6 kyu**

**Palindromization**

26691% of 9441 of323[Detecter](https://www.codewars.com/users/Detecter)

C#

* [TRAIN AGAIN](https://www.codewars.com/kata/palindromization/train/csharp)
* [NEXT KATA](https://www.codewars.com/trainer/csharp)

Details

[Solutions](https://www.codewars.com/kata/palindromization/solutions/csharp)

[Forks (1)](https://www.codewars.com/kata/palindromization/forks/csharp)

[Discourse (12)](https://www.codewars.com/kata/palindromization/discuss/csharp)

* Add to Collection
* |
* Share this kata:

You are given string "elements" and an int "n". Your task is to return a string that is a palindrom using elements from the string "elements" with the length "n".

The format of the palindromization:

* Your palindrome begins with the first element of "elements".
* After obtaining a pair, you insert the next element in "elements" to the palindrome.
* The next element will be paired inside the first pair.
* Repeat
* If you have reached the last element of "elements" then repeat the process from the start.

Error cases:   
If the string elements is empty or n is smaller than 2, return the string "Error!"

Examples:   
<<<<<<< mine

For elements = "123"

n = 2 => result = "11"

n = 3 => result = "121"

n = 4 => result = "1221"

n = 5 => result = "12321"

n = 6 => result = "123321"

n = 7 => result = "1231321"

n = 8 => result = "12311321"

n = 9 => result = "123121321"

n = 10=> result = "1231221321"

<https://www.codewars.com/kata/palindromization/csharp>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

public static string Palindromization(string elements, int n)

{

if (elements.Length ==0 || n < 2) return "Error!";

int t = (int)Math.Ceiling(n / 2.0);

//Console.WriteLine("n " + t);

int cuantas = t / elements.Length;

int resto = t % elements.Length;

//Console.WriteLine(cuantas);

//Console.WriteLine(resto);

string a = "";

for(int i =0; i<cuantas; i++)

{

a += elements;

}

string restoString = elements.Substring(0, resto);

char[] rev\_a = a.ToCharArray();

Array.Reverse(rev\_a);

string reverso = new string(rev\_a);

char[] rev\_resto = restoString.ToCharArray();

Array.Reverse(rev\_resto);

string rev\_restoString = new string(rev\_resto);

string concat = a + restoString + rev\_restoString + reverso;

if (concat.Length > n)

{

if(rev\_restoString.Length >= 1)

rev\_restoString = rev\_restoString.Substring(1);

else

{

reverso = reverso.Substring(1);

}

concat = a + restoString + rev\_restoString + reverso;

return concat;

}

return concat;

}

static void Main(string[] args)

{

string s = "123";

int tam = 1;

Console.WriteLine(Palindromization(s, 5));

//for (int i = 1; i <= 20; i++)

//{

// Console.WriteLine(i + " " + Palindromization(s, i) + " "

// + Palindromization(s, i).Length);

//}

//int n = (int)Math.Ceiling(tam / 2.0);

//Console.WriteLine("n " + n);

//int cuantas = n / s.Length;

//int resto = n % s.Length;

//Console.WriteLine(cuantas);

//Console.WriteLine(resto);

/\*

For elements = "123"

n = 2 => result = "11"

n = 3 => result = "121"

n = 4 => result = "1221"

n = 5 => result = "12321"

n = 6 => result = "123321"

n = 7 => result = "1231321"

n = 8 => result = "12311321"

n = 9 => result = "123121321"

n = 10=> result = "1231221321"

\*/

Console.ReadLine();

}

}

}