**5 kyu**

**Going to zero or to infinity?**

2832985% of 321200 of2,013[g964](https://www.codewars.com/users/g964" \o "This kata's Sensei)

C#

* [TRAIN AGAIN](https://www.codewars.com/kata/going-to-zero-or-to-infinity/train/csharp)
* [NEXT KATA](https://www.codewars.com/trainer/csharp)

Details

[Solutions](https://www.codewars.com/kata/going-to-zero-or-to-infinity/solutions/csharp)

[Forks (1)](https://www.codewars.com/kata/going-to-zero-or-to-infinity/forks/csharp)

[Discourse (157)](https://www.codewars.com/kata/going-to-zero-or-to-infinity/discuss/csharp)

* Add to Collection
* |
* Share this kata:

Consider the following numbers (where n! is factorial(n)):

u1 = (1 / 1!) \* (1!)

u2 = (1 / 2!) \* (1! + 2!)

u3 = (1 / 3!) \* (1! + 2! + 3!)

un = (1 / n!) \* (1! + 2! + 3! + ... + n!)

Which will win: 1 / n! or (1! + 2! + 3! + ... + n!)?

Are these numbers going to 0 because of 1/n! or to infinity due to the sum of factorials?

Task

Calculate (1 / n!) \* (1! + 2! + 3! + ... + n!) for a given n, where n is an integer greater or equal to 1.

To avoid discussions about rounding, return the result **truncated** to 6 decimal places, for example:

1.0000989217538616 will be truncated to 1.000098

1.2125000000000001 will be truncated to 1.2125

Remark

Keep in mind that factorials grow rather rapidly, and you need to handle large inputs.

<https://www.codewars.com/kata/going-to-zero-or-to-infinity/csharp>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Numerics;

using System.Globalization;

namespace ConsoleApp1

{

class Program

{

public class Suite

{

public static double going(int n)

{

double running = 1.0, term = 1.0;

for (int i = n; i > 1; i--) running += (term \*= 1.0 / i);

return System.Math.Truncate(running \* 1000000) / 1000000;

}

}

public class Suite

{

public static double going(int n)

{

double res = 1.0; double inter = 1.0;

for (int i = n; i >= 2; i--)

{

inter = inter \* (1.0 / i);

res += inter;

}

return Math.Floor(res \* Math.Pow(10, 6)) / Math.Pow(10, 6);

}

}

//--------------NO FUNCIONA -------------------

public static decimal TruncateDecimal(decimal value, int precision)

{

decimal step = (decimal)Math.Pow(10, precision);

decimal tmp = Math.Truncate(step \* value);

return tmp / step;

}

public static double going(int n)

{

//BigInteger bigInteger = new BigInteger();

// your code

double[] hash = new double[n + 1];

hash[1] = 1.0D;

for (int i = 2; i <= n; i++)

{

hash[i] = i \* hash[i - 1];

}

double a = 1.0D / hash[n];

double b = 0.0D;

for (int i = 1; i <= n; i++)

{

b += hash[i] ;

}

//return double.Parse( (a \* b).ToString("0.00000") );

var culture = new CultureInfo("en-US");

string s = (a \* b).ToString();

//if (s == "0") s = "1";

return Convert.ToDouble( TruncateDecimal(decimal.Parse(s, culture), 6));

}

static void Main(string[] args)

{

//Assert.AreEqual(1.275, Suite.going(5));

Console.WriteLine(going(5));

//double value = 0.05666666666;

//double res = Math.Truncate(100 \* value) / 100;

//Console.WriteLine(res );

Console.ReadLine();

}

}

}