1 uzduotis – Sum of Negatives

Uzduotis-

Create a function that takes an array of positive and negative numbers. Return an array where the first element is the **count** of positive numbers and the second element is the **sum** of negative numbers.

Examples

```
CountPosSumNeg([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, -11, -12, -13, -14, -15]) → [
// There are a total of 10 positive numbers.
// The sum of all negative numbers equals -65.

CountPosSumNeg([92, 6, 73, -77, 81, -90, 99, 8, -85, 34]) → [7, -252]

CountPosSumNeg([91, -4, 80, -73, -28]) → [2, -105]

CountPosSumNeg([]) → []
```

Notes

- If given an empty array, return an empty array: `[]`
- Cast sum to int, don't mind the remaining decimal places.
- 0 is not positive.

SUGGEST EDIT

C# source kodas –

```
public class Program
{
    public static int[] CountPosSumNeg(double[] arr)
    {
      if(arr.Length == 0 || arr == null)
      {
        return new int[2];
      }
      int[] result = new int[] {0, 0};
      for(int i = 0; i < arr.Length; i++)
      {
        if(arr[i] > 0)
        {
          result[0]++;
        }
        else
        {
          result[1] += (int)arr[i];
        }
      }

      return result;
    }
}
```

Ruby kodas :

```ruby
#Sukuriama nauja Sum klase, kurios viduje deklaruotas self metodas
class Sum
  #Sukuriamas naujas klases metodas, kuris apskaiciuoja neigiamu skaiciu suma, Skaicius gauna array pavidalu.
  def self.SumOfNegatives(array)
    #Per kiekviena array elementa sukamas for ciklas ir tikrinama ar skaicius i yra teigiamas arba neigiamas
    #Jei skaicius neigiamas yra pridedamas prie atsakymo sumos
   for i in array
     if i<0
       sum=sum.to_i+i.to_i
     end
    end
    #Atspausdina atsakyma ekrane
  puts sum
  end
end
array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, -11, -12, -13, -14, -15]
#Kreipinys i sum klases self SumOfNegatives metoda paduodant array
Sum.SumOfNegatives(array)
```

Console:

```
> ruby main.rb
-65
>
```

2 uzduotis – Array of multiples

Create a function that takes two numbers as arguments (`num`, `length`) and returns an array of multiples of `num` up to `length`.

Examples

```
ArrayOfMultiples(7, 5) → [7, 14, 21, 28, 35]

ArrayOfMultiples(12, 10) → [12, 24, 36, 48, 60, 72, 84, 96, 108, 120]

ArrayOfMultiples(17, 6) → [17, 34, 51, 68, 85, 102]
```

Notes

Notice that `num` is also included in the returned array.

SUGGEST EDIT

Source kodas c#

```
using System;

public class Program {
    public static int[] ArrayOfMultiples(int num, int length)
    {
        int[] multiples = new int[length];

        for(int i = 0; i < length; i++) {
            multiples[i] = num * (i + 1);
        }
        return multiples;
    }
}
```

Ruby kodas

second.rb

```ruby
class ArrayOfMultiples
  def self.multiply(firstNr, secondNr)
    answer = []
    sum = 0
    i = 0
    while i < firstNr.to_i do
  sum+=secondNr.to_i
  answer.push(sum)
  i += 1
  end
  puts "Array of multiples:"
  puts answer
  end
end



require "readline"
puts "Input first number:"
firstNr = gets
puts "Input second number:"
secondNr = gets
ArrayOfMultiples.multiply(firstNr, secondNr)
```

Console :

```
> ruby main.rb
Input first number:
5
Input second number:
7
Array of multiples:
7
14
21
28
35
>
```

3 uzduotis

Uzduotis :

Create a function that returns `true` if a number is a palindrome.

Examples

```
IsPalindrome(838) → true

IsPalindrome(4433) → false

IsPalindrome(443344) → true
```

Notes

- A palindrome is a number that remains the same when reversed.
- Bonus: Try solving this without turning the number into a string.

C# source kodas:

```
public class Program
{
    public static bool IsPalindrome(int num)
    {
      string str = num.ToString();
      for (int i = 0; i < str.Length / 2; i++)
    {
        if (str[i] != str[str.Length - i - 1])
            return false;
    }
    return true;
    }
}
```

Ruby codas:

main.rb

```ruby
class Number
  def self.Polyndrome(firstNr)
  reversed =0
temp = firstNr.to_i

while firstNr.to_i > 0 do
    reversed = reversed *10
    reversed = reversed + (firstNr.to_i%10)
    firstNr = firstNr.to_i/10
end
PutAnswer(reversed, temp)
end
class PrintInfo
  def self.PutAnswer(reversed,temp)
  if reversed==temp
    puts "palindromas"
 else
  puts "ne palindromas"
      end
  end
end
end


require "readline"
firstNr=0
puts "Input number:"
firstNr = gets
Number.Polyndrome(firstNr)
```

Console:

```
Input number:
15151
palindromas
```

```
ruby main.rb
Input number:
25555566
ne palindromas
```

4uzduotis –

Uzduotis -

Create a Fact method that receives a **non-negative** integer and `returns` the factorial of that number.

Examples

```
Fact(1) → 1

Fact(3) → 6

Fact(6) → 720
```

Notes

- Consider that 0! = 1.
- You should return a `long` data type number.

C# source kodas

```
public class Program
{
  public static long Fact(int n)
    => n <= 1 ? 1 : n * Fact(n - 1);
}
```

Ruby codas

main.rb

```ruby
class Integer
  def fact
    (1..self).reduce(:*) || 1
  end
end

require "readline"
firstNr=0
puts "Input number:"
firstNr = gets


puts firstNr.to_i.fact

```

Console:

```
> ruby main.rb
Input number:
10
3628800
>
```