Moves in squared strings

<https://www.codewars.com/kata/moves-in-squared-strings-i/train/csharp>

This kata is the first of a sequence of four about "Squared Strings".

You are given a string of n lines, each substring being n characters long: For example:

s = "abcd\nefgh\nijkl\nmnop"

We will study some transformations of this square of strings.

* Vertical mirror: vert\_mirror (or vertMirror or vert-mirror)

vert\_mirror(s) => "dcba\nhgfe\nlkji\nponm"

* Horizontal mirror: hor\_mirror (or horMirror or hor-mirror)

hor\_mirror(s) => "mnop\nijkl\nefgh\nabcd"

or printed:

vertical mirror |horizontal mirror abcd --> dcba |abcd --> mnop efgh hgfe |efgh ijkl ijkl lkji |ijkl efgh mnop ponm |mnop abcd

#Task:

* Write these two functions

and

* high-order function oper(fct, s) where
  + fct is the function of one variable f to apply to the string s (fct will be one of vertMirror, horMirror)

#Examples:

s = "abcd\nefgh\nijkl\nmnop" oper(vert\_mirror, s) => "dcba\nhgfe\nlkji\nponm" oper(hor\_mirror, s) => "mnop\nijkl\nefgh\nabcd"

**Note:**

The form of the parameter fct in oper changes according to the language. You can see each form according to the language in "Sample Tests".

using System;

using System.Linq;

using NUnit.Framework;

public class Opstrings

{

public static string VertMirror(string strng)

{

// your code

var subStrings = strng.Split('\n');

var resultString = "";

for(var i = 0; i < subStrings.Length; i++)

{

var subString = subStrings[i];

string mirror = "";

var characters = subString.ToCharArray();

for(int j = characters.Length - 1; j >= 0; j--)

{

mirror += characters[j];

}

resultString += mirror;

if(i < subStrings.Length - 1)

{

resultString += '\n';

}

}

return resultString;

}

public static string HorMirror(string strng)

{

var subStrings = strng.Split('\n');

var resultString = "";

for(int i = subStrings.Length - 1; i >= 0; i--)

{

resultString += subStrings[i];

if(i > 0)

{

resultString += '\n';

}

}

return resultString;

}

public static string Oper(Func<string, string> fct, string s)

{

// your code and complete ... before operator

return fct(s);

}

}