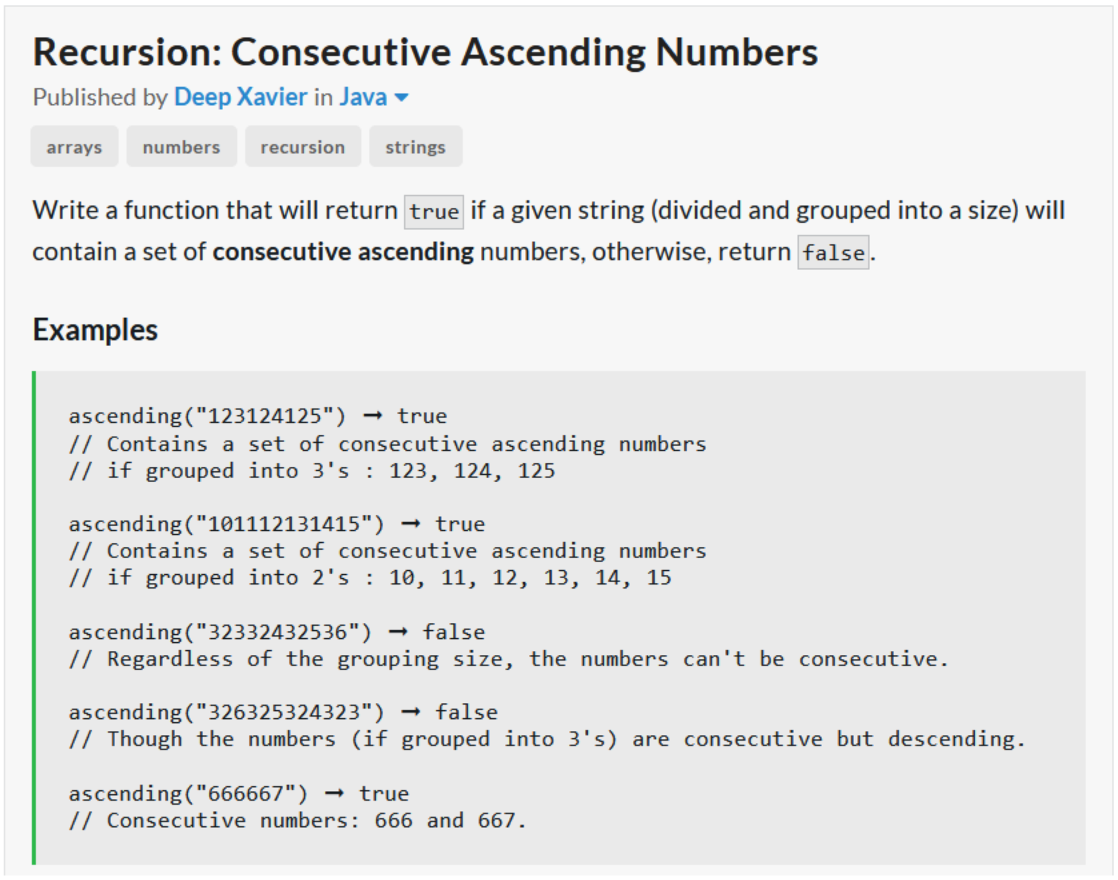


Recursion: Consecutive Ascending Numbers

Published by Deep Xavier in Java

arrays numbers recursion strings

Write a function that will return `true` if a given string (divided and grouped into a size) will contain a set of **consecutive ascending** numbers, otherwise, return `false`.

Examples

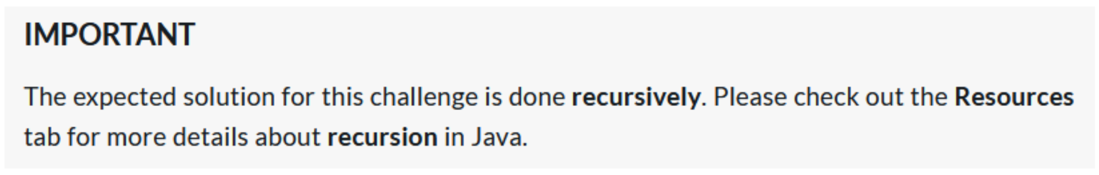
```
ascending("123124125") ➞ true
// Contains a set of consecutive ascending numbers
// if grouped into 3's : 123, 124, 125

ascending("101112131415") ➞ true
// Contains a set of consecutive ascending numbers
// if grouped into 2's : 10, 11, 12, 13, 14, 15

ascending("32332432536") ➞ false
// Regardless of the grouping size, the numbers can't be consecutive.

ascending("326325324323") ➞ false
// Though the numbers (if grouped into 3's) are consecutive but descending.

ascending("666667") ➞ true
// Consecutive numbers: 666 and 667.
```



IMPORTANT

The expected solution for this challenge is done **recursively**. Please check out the **Resources** tab for more details about **recursion** in Java.

n would be length of characters (size of the groups) to be examined.
it will start with 1. It would be declared as a static variable

It appears that first area to check would be to perform Str.length()%n ==0 within ascending("XXXXXXXXXX")
This will ensure that the divided block is viable.
Otherwise perform n=n+1 and ascending("XXXXXXXXXX")

Also,

if str.length()==0
{
Can attempt to deduce if it is consecutive ascending numbers.
It would compare str.indexOf(n-1) with str.indexOf(n)
if long.valueOf(str.indexOf(n-1) < long.valueOf(str.indexOf(n)

It is consecutive ascending order numbers

return true;

Note this notation indexOf is successful if n=1
However need to examine broader perspective of substring with n being greater than 1

if long.valueOf(str.substring(0,n) < long.valueOf(str.substring(n,(2*n))
It is consecutive ascending order numbers since it would have processed all the blocks in question.

return true;

}

else //str.length()>0
{

it will perform str.substring(str.length()-n, str.length()) to get the last block
store in variable String block;

try
{
it will perform str.substring(str.length()- (2xn), str.length()-n) to get the block before last of size n
store in variable String blockBefore;

if long.valueOf(blockBefore) < long.valueOf(block)
{
this is ascending
perform recursion call ascending("XXXXXXXXXX");

}

else
{
this is descending
this would also perform recursion call
But this time we have to increase the size of the width of the integers n=n+1;
}

```
} //end try


/*
catch
{
//NOTE: if it enters here, we know there are odd number of groups
I do not believe there is anything suitable here, since the logic here is being with in
if str.length()==0
Perhaps this block can be replaced with finally
}
*/

finally
{
Not entire sure what to include in here at the moment
return false;
}
```