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.

It appears that first area would be to perform Str.length()%n ==0
if true, then perform (This will ascertain if the challenge is compliant without any indexing exceptions)
Now, we do not need to store each consecutive number / segments of n characters into an array numbers of size (str.length()/n).
If this is undertaken, it will violate recursion since an attempt will be made to traverse it iteratively...

Instead:

```
if (str.length()!=0)
{
perform str.substring(str.length()-3, str.length())
}

else
{
```

Can deduce if it is consecutive ascending numbers.
}

Note: This would allow it to process 3 characters from the String (and store in temp variable).
It would perform Long.valueOf(temp).
It would store the value in numbers[pos]
It would need to set up a variable such as pos. This would alternate between index 0 and 1 of the numbers array. Strictly speaking, we are staying away from iteration...

We would need logic as follows before the recursive call
if pos==1
pos=0

if pos==0
pos=1

Perform recursion call (str.substring(0,str.length()-3), numbers[pos])
This has hit the compromise of the recursive call.
We are now operating with two arguments.
It is totally unclear if this is permitted.
For the moment, I will continue.

It would now perform exactly same as above...

if (str.length()!=0)
{
perform str.substring(str.length()-3, str.length())
}

else
{
Can deduce if it is consecutive ascending numbers by comparing numbers[0] and numbers[1]

We also need to be careful since if the numbers being compared were
121, 122,123
It would have stored the numbers as follows:
numbers[1] = 123
numbers[0] = 122

numbers[1] = 121

It would have clearly made a false comparison against numbers[0] since it has not been populated.

We can not assume that a 0 is the default value of numbers[0].
Since it might also be a genuine value in the string in ascending function

}

It would store this value in an array. The size of the array would be determined by