n-connected

A non empty String is said to be n-connected if it contains n consecutive characters that are the same. For example, the number “ABCDEF” is 1-connected, “AABCDEF” is a 2-connected and "AABBBCDEF" is a 3-connected. The minimum n value for a n-connected String is 1.

* You may assume all Strings in this problem will be non empty and contain only UPPER case letters!

In this problem you will implement two static methods in nConnected class.

The first method is the int getNConnected(String str) which return the n-Connected value of the parameter str. You may assume str != null.

The following code shows the results of the getNConnected method.

|  |  |
| --- | --- |
| The following code | Returns |
| nConnected.getNConnected("ABCDEF"); | 1 |
| nConnected.getNConnected("AABCDEF"); | 2 |
| nConnected.getNConnected("AABBBCDBBEF") | 3 |
| nConnected.getNConnected("AAABCCCCDCCCEF"); | 4 |

The second method is the String rotateKitems(String str, int k). This method will find the smallest String with the largest n-connected String that can be formed by any number of iterations of removing the first k elements of the parameter str and concatenating those k elements to the end of the String. For example, rotateKitems(“TEST”, 3) will search the following Strings: “TEST”, “TTES”, “STTE”, and “ESTT”. The next String in the sequence is “TEST” which started the sequence. “TTES”, “STTE”, and “ESTT” all have an n-connected value of 2, therefore “ESTT” is returned since “ESTT”.compareTo(“TTES”) < 0 and “ESTT”.compareTo(“STTE”) < 0.

The following code shows the results of the getNConnected method.

|  |  |
| --- | --- |
| The following code | Returns |
| nConnected.rotateKitems("TEST", 3); | "ESTT" |
| nConnected.rotateKitems("TEST", 2); | "STTE" |
| nConnected.rotateKitems("RABBBCDEF", 2); | "ABBBCDEFR" |
| nConnected.rotateKitems("EAFBEE", 2)); | “EEEAFB” |