Given a non-negative int n, compute recursively (no loops) the count of the occurrences of 8 as a digit, except that an 8 with another 8 immediately to its left counts double, so 8818 yields 4. Note that mod (%) by 10 yields the rightmost digit (126 % 10 is 6), while divide (/) by 10 removes the rightmost digit (126 / 10 is 12).

count8(8) → 1
count8(818) → 2
count8(8818) → 4

public static int count8(int n)

Given a string, compute recursively (no loops) the number of times lowercase "hi" appears in the string.

countHi("xxhixx") → 1
countHi("xhixhix") → 2
countHi("hi") → 1

public static int countHi(String str)

Given a string, compute recursively the number of times lowercase "hi" appears in the string, however do not count "hi" that have an 'x' immedately before them.

countHi2("ahixhi") → 1
countHi2("ahibhi") → 2
countHi2("xhixhi") → 0

public static int countHi2(String str)

Given a string and a non-empty substring **sub**, compute recursively the number of times that sub appears in the string, without the sub strings overlapping.

strCount("catcowcat", "cat") → 2
strCount("catcowcat", "cow") → 1
strCount("catcowcat", "dog") → 0

public static int strCount(String str, String sub)

Given a string, return recursively a "cleaned" string where adjacent chars that are the same have been reduced to a single char. So "yyzzza" yields "yza".

stringClean("yyzzza") → "yza"
stringClean("abbbcdd") → "abcd"
stringClean("Hello") → "Helo"

public static String stringClean(String str)