Доброе день, String - дос

1. Given a string, return a string where for every char in the original, there are two chars.

```
doubleChar("The") → "TThhee"
doubleChar("AAbb") → "AAAAbbbb"
doubleChar("Hi-There") → "HHii--TThheerree"
```

2. Return the number of times that the string "code" appears anywhere in the given string, except we'll accept any letter for the 'd', so "cope" and "cooe" count.

```
countCode("aaacodebbb") → 1
countCode("codexxcode") → 2
countCode("cozexxcope") → 2
```

3. Return true if the given string contains a "bob" string, but where the middle 'o' char can be any char.

```
bobThere("abcbob") → true
bobThere("b9b") → true
bobThere("bac") → false
```

4. Given a string and an int n, return a string made of n repetitions of the last n characters of the string. You may assume that n is between 0 and the length of the string, inclusive.

```
repeatEnd("Hello", 3) → "llollollo"
repeatEnd("Hello", 2) → "lolo"
repeatEnd("Hello", 1) → "o"
```

5. Given a string, consider the prefix string made of the first N chars of the string. Does that prefix string appear somewhere else in the string? Assume that the string is not empty and that N is in the range 1..str.length().

```
prefixAgain("abXYabc", 1) → true
prefixAgain("abXYabc", 2) → true
prefixAgain("abXYabc", 3) → false
```

6. Returns true if for every '*' (star) in the string, if there are chars both immediately before and after the star, they are the same.

```
sameStarChar("xy*yzz") → true
sameStarChar("xy*zzz") → false
sameStarChar("*xa*az") → true
```

7. Return a version of the given string, where for every star (*) *in the string the star and the chars immediately to its left and right are gone. So "ab*cd" yields "ad" and "ab**cd" also yields "ad".*

```

```

```
starOut("ab*cd") → "ad"
starOut("ab**cd") → "ad"
starOut("sm*eilly") → "silly"
```

8. Return the number of times that the string "hi" appears anywhere in the given string.

```
countHi("abc hi ho") → 1
countHi("ABChi hi") → 2
countHi("hihi") → 2
```

9. Given two strings, return true if either of the strings appears at the very end of the other string, ignoring upper/lower case differences (in other words, the computation should not be "case sensitive"). Note: str.toLowerCase() returns the lowercase version of a string.

```
endOther("Hiabc", "abc") → true
endOther("AbC", "HiaBc") → true
endOther("abc", "abXabc") → true
```

10. We'll say that a String is xy-balanced if for all the 'x' chars in the string, there exists a 'y' char somewhere later in the string. So "xxy" is balanced, but "xyx" is not. One 'y' can balance multiple 'x's. Return true if the given string is xy-balanced.

```
xyBalance("aaxbby") → true
xyBalance("aaxbb") → false
xyBalance("yaaxbb") → false
```

11. Given a string and an int n, return a string made of the first n characters of the string, followed by the first n-1 characters of the string, and so on. You may assume that n is between 0 and the length of the string, inclusive (i.e. n >= 0 and n <= str.length()).

```
repeatFront("Chocolate", 4) → "ChocChoChC"
repeatFront("Chocolate", 3) → "ChoChC"
repeatFront("Ice Cream", 2) → "IcI"
```

12. Given a string, does "xyz" appear in the middle of the string? To define middle, we'll say that the number of chars to the left and right of the "xyz" must differ by at most one. This problem is harder than it looks.

```
xyzMiddle("AAxyzBB") → true
xyzMiddle("AxyzBB") → true
xyzMiddle("AxyzBBB") → false
```

13. Given a string, compute a new string by moving the first char to come after the next two chars, so "abc" yields "bca". Repeat this process for each subsequent group of 3 chars, so "abcdef" yields "bcaefd". Ignore any group of fewer than 3 chars at the end.

```
oneTwo("abc") → "bca"
oneTwo("tca") → "cat"
oneTwo("tcagdo") → "catdog"
```

14. Given a string and a non-empty word string, return a version of the original String where all chars have been replaced by pluses ("+"), except for appearances of the word string which are preserved unchanged.

```
plusOut("12xy34", "xy") → "++xy++"
plusOut("12xy34", "1") → "1+++++"
plusOut("12xy34xyabcxy", "xy") → "++xy++xy+++xy"
```

15. Return true if the string "cat" and "dog" appear the same number of times in the given string.

```
catDog("catdog") → true
catDog("catcat") → false
catDog("1cat1cadodog") → true
```

16. Return true if the given string contains an appearance of "xyz" where the xyz is not directly preceeded by a period (.). So "xxyz" counts but "x.xyz" does not.

```
xyzThere("abcxyz") → true
xyzThere("abc.xyz") → false
xyzThere("xyz.abc") → true
```

17. Given two strings, a and b, create a bigger string made of the first char of a, the first char of b, the second char of a, the second char of b, and so on. Any leftover chars go at the end of the result.

```
mixString("abc", "xyz") → "axbycz"
mixString("Hi", "There") → "HTihere"
mixString("xxxx", "There") → "xTxhxexre"
```

18. Given two strings, word and a separator sep, return a big string made of count occurrences of the word, separated by the separator string.

```
repeatSeparator("Word", "X", 3) → "WordXWordXWord"
repeatSeparator("This", "And", 2) → "ThisAndThis"
repeatSeparator("This", "And", 1) → "This"
```

19. A sandwich is two pieces of bread with something in between. Return the string that is between the first and last appearance of "bread" in the given string, or return the empty string "" if there are not two pieces of bread.

```
getSandwich("breadjambread") → "jam"
getSandwich("xxbreadjambreadyy") → "jam"
getSandwich("xxbreadyy") → ""
```

20. Look for patterns like "zip" and "zap" in the string -- length-3, starting with 'z' and ending with 'p'. Return a string where for all such words, the middle letter is gone, so "zipXzap" yields "zpXzp".

```
zipZap("zipXzap") → "zpXzp"
zipZap("zopzop") → "zpzp"
zipZap("zzzopzop") → "zzzpzp"
```

21. Given a string and a non-empty word string, return a string made of each char just before and just after every appearance of the word in the string. Ignore cases where there is no char before or after the word, and a char may be included twice if it is between two words.

```
wordEnds("abcXY123XYijk", "XY") → "c13i"
wordEnds("XY123XY", "XY") → "13"
wordEnds("XY1XY", "XY") → "11"
```