Добрый вечер, String - трес

1. Given a string, count the number of words ending in 'y' or 'z' -- so the 'y' in "heavy" and the 'z' in "fez" count, but not the 'y' in "yellow" (not case sensitive). We'll say that a y or z is at the end of a word if there is not an alphabetic letter immediately following it. (Note: Character.isLetter(char) tests if a char is an alphabetic letter.)

```
countYZ("fez day") → 2
countYZ("day fez") → 2
countYZ("day fyyyz") → 2
```

2. We'll say that a lowercase 'g' in a string is "happy" if there is another 'g' immediately to its left or right. Return true if all the g's in the given string are happy.

```
gHappy("xxggxx") → true
gHappy("xxgxx") → false
gHappy("xxggyygxx") → false
```

3. Given a string, return the longest substring that appears at both the beginning and end of the string without overlapping. For example, sameEnds("abXab") is "ab".

```
sameEnds("abXYab") → "ab"
sameEnds("xx") → "x"
sameEnds("xxx") → "x"
```

4. Given a string, return the sum of the numbers appearing in the string, ignoring all other characters. A number is a series of 1 or more digit chars in a row. (Note: Character.isDigit(char) tests if a char is one of the chars '0', '1', .. '9'. Integer.parseInt(string) converts a string to an int.)

```
sumNumbers("abc123xyz") → 123
sumNumbers("aa11b33") → 44
sumNumbers("7 11") → 18
```

5. Given two strings, base and remove, return a version of the base string where all instances of the remove string have been removed (not case sensitive). You may assume that the remove string is length 1 or more. Remove only non-overlapping instances, so with "xxx" removing "xx" leaves "x".

```
withoutString("Hello there", "llo") → "He there"
withoutString("Hello there", "e") → "Hllo thr"
withoutString("Hello there", "x") → "Hello there"
```

6. We'll say that a "triple" in a string is a char appearing three times in a row. Return the number of triples in the given string. The triples may overlap.

```
countTriple("abcXXXabc") → 1
countTriple("xxxabyyyycd") → 3
countTriple("a") → 0
```

7. Given a string, look for a mirror image (backwards) string at both the beginning and end of the given string. In other words, zero or more characters at the very begining of the given string, and at the very end of the string in reverse order (possibly overlapping). For example, the string "abXYZba" has the mirror end "ab".

```
mirrorEnds("abXYZba") → "ab"
mirrorEnds("abca") → "a"
mirrorEnds("aba") → "aba"
```

8. Given a string, return a string where every appearance of the lowercase word "is" has been replaced with "is not". The word "is" should not be immediately preceeded or followed by a letter -- so for example the "is" in "this" does not count. (Note: Character.isLetter(char) tests if a char is a letter.)

```
notReplace("is test") → "is not test"
notReplace("is-is") → "is not-is not"
notReplace("This is right") → "This is not right"
```

9. Given a string, return true if the number of appearances of "is" anywhere in the string is equal to the number of appearances of "not" anywhere in the string (case sensitive).

```
equalIsNot("This is not") → false
equalIsNot("This is notnot") → true
equalIsNot("noisxxnotyynotxisi") → true
```

10. Given a string, return the sum of the digits 0-9 that appear in the string, ignoring all other characters. Return 0 if there are no digits in the string. (Note: Character.isDigit(char) tests if a char is one of the chars '0', '1', .. '9'. Integer.parseInt(string) converts a string to an int.)

```
sumDigits("aa1bc2d3") → 6
sumDigits("aa11b33") → 8
sumDigits("Chocolate") → 0
```

11. Given a string, return the length of the largest "block" in the string. A block is a run of adjacent chars that are the same.

```
maxBlock("hoopla") → 2
maxBlock("abbCCCddBBBxx") → 3
maxBlock("") → 0
```