Доброе утро, String - уно

1. Given a string name, e.g. "Bob", return a greeting of the form "Hello Bob!".

```
helloName("Bob") → "Hello Bob!"
helloName("Alice") → "Hello Alice!"
helloName("X") → "Hello X!"
```

2. Given an "out" string length 4, such as "<<>>", and a word, return a new string where the word is in the middle of the out string, e.g. "<>". Note: use str.substring(i, j) to extract the String starting at index i and going up to but not including index j.

```
makeOutWord("<<>>", "Yay") → "<<Yay>>"
makeOutWord("<<>>", "WooHoo") → "<<WooHoo>>"
makeOutWord("[[]]", "word") → "[[word]]"
```

3. Given a string of even length, return the first half. So the string "WooHoo" yields "Woo".

```
firstHalf("WooHoo") → "Woo"
firstHalf("HelloThere") → "Hello"
firstHalf("abcdef") → "abc"
```

4. Given 2 strings, return their concatenation, except omit the first char of each. The strings will be at least length 1.

```
nonStart("Hello", "There") → "ellohere"
nonStart("java", "code") → "avaode"
nonStart("shotl", "java") → "hotlava"
```

5. Given a string, return a string length 1 from its front, unless front is false, in which case return a string length 1 from its back. The string will be non-empty.

```
theEnd("Hello", true) → "H"
theEnd("Hello", false) → "o"
theEnd("oh", true) → "o"
```

6. Given a string, return true if it ends in "ly".

```
endsLy("oddly") → true
endsLy("y") → false
endsLy("oddy") → false
```

7. Given a string of odd length, return the string length 3 from its middle, so "Candy" yields "and". The string length will be at least 3.

```
middleThree("Candy") → "and"
middleThree("and") → "and"
middleThree("solving") → "lvi"
```

8. Given 2 strings, a and b, return a new string made of the first char of a and the last char of b, so "yo" and "java" yields "ya". If either string is length 0, use '@' for its missing char.

```
lastChars("last", "chars") → "ls"
lastChars("yo", "java") → "ya"
lastChars("hi", "") → "h@"
```

9. Given a string, if the string begins with "red" or "blue" return that color string, otherwise return the empty string.

```
seeColor("redxx") → "red"
seeColor("xxred") → ""
seeColor("blueTimes") → "blue"
```

10. Given a string, return a new string made of 3 copies of the first 2 chars of the original string. The string may be any length. If there are fewer than 2 chars, use whatever is there.

```
extraFront("Hello") → "HeHeHe"
extraFront("ab") → "ababab"
extraFront("H") → "HHH"
```

11. Given a string and a second "word" string, we'll say that the word matches the string if it appears at the front of the string, except its first char does not need to match exactly. On a match, return the front of the string, or otherwise return the empty string. So, so with the string "hippo" the word "hi" returns "hi" and "xip" returns "hip". The word will be at least length 1.

```
startWord("hippo", "hi") → "hi"
startWord("hippo", "xip") → "hip"
startWord("hippo", "i") → "h"
```

12. Given two strings, a and b, return the result of putting them together in the order abba, e.g. "Hi" and "Bye" returns "HiByeByeHi".

```
makeAbba("Hi", "Bye") → "HiByeByeHi"
makeAbba("Yo", "Alice") → "YoAliceAliceYo"
makeAbba("What", "Up") → "WhatUpUpWhat"
```

13. Given a string, return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2.

```
extraEnd("Hello") → "lololo"
extraEnd("ab") → "ababab"
extraEnd("Hi") → "HiHiHi"
```

14. Given a string, return a version without the first and last char, so "Hello" yields "ell". The string length will be at least 2.

```
withoutEnd("Hello") → "ell"
withoutEnd("java") → "av"
withoutEnd("coding") → "odin"
```

15. Given a string, return a "rotated left 2" version where the first 2 chars are moved to the end. The string length will be at least 2.

```
left2("Hello") → "lloHe"
left2("java") → "vaja"
left2("Hi") → "Hi"
```

16. Given a string, return a version without both the first and last char of the string. The string may be any length, including 0.

```
withoutEnd2("Hello") → "ell"
withoutEnd2("abc") → "b"
withoutEnd2("ab") → ""
```

17. Given a string and an int n, return a string made of the first and last n chars from the string. The string length will be at least n.

```
nTwice("Hello", 2) → "Helo"
nTwice("Chocolate", 3) → "Choate"
nTwice("Chocolate", 1) → "Ce"
```

18. Given a string, return true if "bad" appears starting at index 0 or 1 in the string, such as with "badxxx" or "xbadxx" but not "xxbadxx". The string may be any length, including 0. Note: use .equals() to compare 2 strings.

```
hasBad("badxx") → true
hasBad("xbadxx") → true
hasBad("xxbadxx") → false
```

19. Given two strings, append them together (known as "concatenation") and return the result. However, if the concatenation creates a double-char, then omit one of the chars, so "abc" and "cat" yields "abcat".

```
conCat("abc", "cat") → "abcat"
conCat("dog", "cat") → "dogcat"
conCat("abc", "") → "abc"
```

20. Given a string, return true if the first 2 chars in the string also appear at the end of the string, such as with "edited".

```
frontAgain("edited") → true
frontAgain("edit") → false
frontAgain("ed") → true
```

21. Given a string, if a length 2 substring appears at both its beginning and end, return a string without the substring at the beginning, so "HelloHe" yields "lloHe". The substring may overlap with itself, so "Hi" yields "". Otherwise, return the original string unchanged.

```
without2("HelloHe") → "lloHe"
without2("HelloHi") → "HelloHi"
without2("Hi") → ""
```

22. Given a string, if the first or last chars are 'x', return the string without those 'x' chars, and otherwise return the string unchanged.

```
withoutX("xHix") → "Hi"
withoutX("xHi") → "Hi"
withoutX("Hxix") → "Hxi"
```

23. The web is built with HTML strings like "*Yay*" which draws Yay as italic text. In this example, the "i" tag makes *and* which surround the word "Yay". Given tag and word strings, create the HTML string with tags around the word, e.g. "*Yay*".

```
makeTags("i", "Yay") → "<i>Yay</i>"
makeTags("i", "Hello") → "<i>Hello</i>"
makeTags("cite", "Yay") → "<cite>Yay</cite>"
```

24. Given a string, return the string made of its first two chars, so the String "Hello" yields "He". If the string is shorter than length 2, return whatever there is, so "X" yields "X", and the empty string "" yields the empty string "". Note that str.length() returns the length of a string.

```
firstTwo("Hello") → "He"
firstTwo("abcdefg") → "ab"
firstTwo("ab") → "ab"
```

25. Given 2 strings, a and b, return a string of the form short+long+short, with the shorter string on the outside and the longer string on the inside. The strings will not be the same length, but they may be empty (length 0).

```
comboString("Hello", "hi") → "hiHellohi"
comboString("hi", "Hello") → "hiHellohi"
comboString("aaa", "b") → "baaab"
```

26. Given a string, return a "rotated right 2" version where the last 2 chars are moved to the start. The string length will be at least 2.

```
right2("Hello") → "loHel"
right2("java") → "vaja"
right2("Hi") → "Hi"
```

27. Given a string of even length, return a string made of the middle two chars, so the string "string" yields "ri". The string length will be at least 2.

```
middleTwo("string") → "ri"
middleTwo("code") → "od"
middleTwo("Practice") → "ct"
```

28. Given a string and an index, return a string length 2 starting at the given index. If the index is too big or too small to define a string length 2, use the first 2 chars. The string length will be at least 2.

```
twoChar("java", 0) → "ja"
twoChar("java", 2) → "va"
twoChar("java", 3) → "ja"
```

29. Given a string, return a string length 2 made of its first 2 chars. If the string length is less than 2, use '@' for the missing chars.

```
atFirst("hello") → "he"
atFirst("hi") → "hi"
atFirst("h") → "h@"
```

30. Given a string of any length, return a new string where the last 2 chars, if present, are swapped, so "coding" yields "codign".

```
lastTwo("coding") → "codign"
lastTwo("cat") → "cta"
lastTwo("ab") → "ba"
```

31. Given two strings, append them together (known as "concatenation") and return the result. However, if the strings are different lengths, omit chars from the longer string so it is the same length as the shorter string. So "Hello" and "Hi" yield "loHi". The strings may be any length.

```
minCat("Hello", "Hi") → "loHi"
minCat("Hello", "java") → "ellojava"
minCat("java", "Hello") → "javaello"
```

32. Given a string, return a version without the first 2 chars. Except keep the first char if it is 'a' and keep the second char if it is 'b'. The string may be any length. Harder than it looks.

```
deFront("Hello") → "llo"
deFront("java") → "va"
deFront("away") → "aay"
```

33. Given a string, if one or both of the first 2 chars is 'x', return the string without those 'x' chars, and otherwise return the string unchanged. This is a little harder than it looks.

```
withoutX2("xHi") → "Hi"
withoutX2("Hxi") → "Hi"
withoutX2("Hi") → "Hi"
```

© - codingbat.com