String-1 > makeOutWord

prev | next | chance

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. "<<word>>". 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]]"

Go ...Save, Compile, Run (ctrl-enter)

```
public String makeOutWord(String out, String word) {
  return out.substring(0,2)+word + out.substring(2);
}
```

Expected	Run		
makeOutWord("<<>>", "Yay") → "<<Yay>>"	"<<Yay>>"	OK	
makeOutWord("<<>>", "WooHoo") → "<<WooHoo>>"	"<<WooHoo>>"	OK	
makeOutWord("[[]]", "word") → "[[word]]"	"[[word]]"	OK	
makeOutWord("HHoo", "Hello") → "HHHellooo"	"HHHellooo"	OK	
makeOutWord("abyz", "YAY") → "abYAYyz"	"abYAYyz"	OK	
other tests		OK	

All Correct

String-1 > nTwice

prev | next | chance

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"

Go ...Save, Compile, Run (ctrl-enter)

```
public String nTwice(String str, int n) {
  return str.substring(0, n) + str.substring(str.length() - n);
}
```

Expected	Run		
nTwice("Hello", 2) → "Helo"	"Helo"	OK	
nTwice("Chocolate", 3) → "Choate"	"Choate"	OK	
nTwice("Chocolate", 1) → "Ce"	"Ce"	OK	
nTwice("Chocolate", 0) → ""	""	OK	
nTwice("Hello", 4) → "Hellello"	"Hellello"	OK	
nTwice("Code", 4) → "CodeCode"	"CodeCode"	OK	
nTwice("Code", 2) → "Code"	"Code"	OK	
other tests		OK	

All Correct

Java | Python

String-1 > makeTags

prev | next | chance

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

Go ...Save, Compile, Run (ctrl-enter)

```
public String makeTags(String tag, String word) {
  return "<"+ tag +">" + word + "<"+ "/"+tag + ">";

}
```

Expected	Run		
makeTags("i", "Yay") → "Yay"	"Yay"	OK	
makeTags("i", "Hello") → "Hello"	"Hello"	OK	
makeTags("cite", "Yay") → "<cite>Yay</cite>"	"<cite>Yay</cite>"	OK	
makeTags("address", "here") → "<address>here</address>"	"<address>here</address>"	OK	
makeTags("body", "Heart") → "<body>Heart</body>"	"<body>Heart</body>"	OK	
makeTags("i", "i") → "i"	"i"	OK	
makeTags("i", "") → ""	""	OK	
other tests		OK	

All Correct

String-1 > firstHalf

prev | next | chance

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"

Go ...Save, Compile, Run (ctrl-enter)

```
public String firstHalf(String str) {
  return str.substring(0,str.length()/2);
}
```

Expected	Run		
firstHalf("WooHoo") → "Woo"	"Woo"	OK	
firstHalf("HelloThere") → "Hello"	"Hello"	OK	
firstHalf("abcdef") → "abc"	"abc"	OK	
firstHalf("ab") → "a"	"a"	OK	
firstHalf("") → ""	""	OK	
firstHalf("0123456789") → "01234"	"01234"	OK	
firstHalf("kitten") → "kit"	"kit"	OK	
other tests		OK	

All Correct

String-1 > nonStart

prev | next | chance

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"

Go ...Save, Compile, Run (ctrl-enter)

```
public String nonStart(String a, String b) {
  return a.substring(1)+b.substring(1);
}
```

Expected	Run		
nonStart("Hello", "There") → "ellohere"	"ellohere"	OK	
nonStart("java", "code") → "avaode"	"avaode"	OK	
nonStart("shotl", "java") → "hotlava"	"hotlava"	OK	
nonStart("ab", "xy") → "by"	"by"	OK	
nonStart("ab", "x") → "b"	"b"	OK	
nonStart("x", "ac") → "c"	"c"	OK	
nonStart("a", "x") → ""	""	OK	
nonStart("kit", "kat") → "itat"	"itat"	OK	
nonStart("mart", "dart") → "artart"	"artart"	OK	
other tests		OK	

All Correct

next | chance

String-1 > theEnd

prev | next | chance

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"

Go ...Save, Compile, Run (ctrl-enter)

```
public String theEnd(String str, boolean front) {
  if(front)
  return str.substring(0,1);
  return str.substring (str.length()-1);
}
```

Expected	Run		
theEnd("Hello", true) → "H"	"H"	OK	
theEnd("Hello", false) → "o"	"o"	OK	
theEnd("oh", true) → "o"	"o"	OK	
theEnd("oh", false) → "h"	"h"	OK	
theEnd("x", true) → "x"	"x"	OK	
theEnd("x", false) → "x"	"x"	OK	
theEnd("java", true) → "j"	"j"	OK	
theEnd("chocolate", false) → "e"	"e"	OK	
theEnd("1234", true) → "1"	"1"	OK	
theEnd("code", false) → "e"	"e"	OK	
other tests		OK	

All Correct

CodingBat code practice

Java | Python

String-1 > makeAbba

prev | next | chance

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"

Go ...Save, Compile, Run (ctrl-enter) Show Hint

```
public String makeAbba(String a, String b) {
 return a+b+b+a;
}
```

Expected	Run	
makeAbba("Hi", "Bye") → "HiByeByeHi"	"HiByeByeHi"	OK
makeAbba("Yo", "Alice") → "YoAliceAliceYo"	"YoAliceAliceYo"	OK
makeAbba("What", "Up") → "WhatUpUpWhat"	"WhatUpUpWhat"	OK
makeAbba("aaa", "bbb") → "aaabbbbbbaaa"	"aaabbbbbbaaa"	OK
makeAbba("x", "y") → "xyyx"	"xyyx"	OK
makeAbba("x", "") → "xx"	"xx"	OK
makeAbba("", "y") → "yy"	"yy"	OK
makeAbba("Bo", "Ya") → "BoYaYaBo"	"BoYaYaBo"	OK
makeAbba("Ya", "Ya") → "YaYaYaYa"	"YaYaYaYa"	OK
other tests		OK

All Correct

String-1 > makeTags

prev | next | chance

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

Go ...Save, Compile, Run (ctrl-enter)

```
public String makeTags(String tag, String word) {
  return "<"+ tag +">" + word + "<"+ "/"+tag + ">";

}
```

Expected	Run	
makeTags("i", "Yay") → "Yay"	"Yay"	OK
makeTags("i", "Hello") → "Hello"	"Hello"	OK
makeTags("cite", "Yay") → "<cite>Yay</cite>"	"<cite>Yay</cite>"	OK
makeTags("address", "here") → "<address>here</address>"	"<address>here</address>"	OK
makeTags("body", "Heart") → "<body>Heart</body>"	"<body>Heart</body>"	OK
makeTags("i", "i") → "i"	"i"	OK
makeTags("i", "") → ""	""	OK
other tests		OK

All Correct

next | chance

String-1 > endsLy

prev | next | chance

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

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

Go ...Save, Compile, Run (ctrl-enter)

```
public boolean endsLy(String str) {
  if(str.length() < 2)
        return false;

    return str.substring(str.length() - 2).equals("ly");
}
```

Expected	Run	
endsLy("oddly") → true	true	OK
endsLy("y") → false	false	OK
endsLy("oddy") → false	false	OK
endsLy("oddl") → false	false	OK
endsLy("olydd") → false	false	OK
endsLy("ly") → true	true	OK
endsLy("") → false	false	OK
endsLy("falsey") → false	false	OK
endsLy("evenly") → true	true	OK
other tests		OK

All Correct

String-1 » middleThree

prev | next | chance

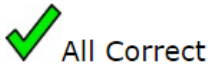
Given a string of odd length, return the string length 3 from its middle, so "Candy" yields "and". The string length will be at least 3.

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

Go ...Save, Compile, Run (ctrl-enter)

```
public String middleThree(String str) {
  int mitad = str.length() / 2;
  return str.substring(mitad - 1, mitad + 2);
}
```

Expected	Run		
middleThree("Candy") → "and"	"and"	OK	
middleThree("and") → "and"	"and"	OK	
middleThree("solving") → "lvi"	"lvi"	OK	
middleThree("Hi yet Hi") → "yet"	"yet"	OK	
middleThree("java yet java") → "yet"	"yet"	OK	
middleThree("Chocolate") → "col"	"col"	OK	
middleThree("XabcxyzabcX") → "xyz"	"xyz"	OK	
other tests		OK	



String-1 » firstTwo

prev | next | chance

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"

Go ...Save, Compile, Run (ctrl-enter)

```
public String firstTwo(String str) {
    if(str.length() < 2)
      return str;

  return str.substring(0, 2);
}
```

Expected	Run		
firstTwo("Hello") → "He"	"He"	OK	
firstTwo("abcdefg") → "ab"	"ab"	OK	
firstTwo("ab") → "ab"	"ab"	OK	
firstTwo("a") → "a"	"a"	OK	
firstTwo("") → ""	""	OK	
firstTwo("Kitten") → "Ki"	"Ki"	OK	
firstTwo("hi") → "hi"	"hi"	OK	
firstTwo("hiya") → "hi"	"hi"	OK	
other tests		OK