|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| String-1 | | | | |
| helloName | makeOutWord | withoutEnd | left2 | theEnd |
| endsLy | middleThree | conCat | seeColor | minCat |
| deFront | withoutX2 |  |  |  |
| String-2 | | | | |
| doubleChar | countHi | repeatSeparator | | getSandwich |

// Given a string name, e.g. "Bob", return a greeting of the form "Hello Bob!".  
 public static String helloName(String name)  
 {

}  
   
 // Given an "out" string with even length, such as "<<>>", and a word,   
 // return a new string where the word is in the middle of the out string, e.g. <<word>>.

// Ex) makeOutWord("<<>>", "WooHoo") -> "<<WooHoo>>"  
 public static String makeOutWord(String out, String word)  
 {

}  
   
 // Given a string, return a version without the first and last char,   
 // so "Hello" yields "ell". The string length will be at least 2.  
 public static String withoutEnd(String str)  
 {

}  
   
 // 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.  
 public static String left2(String str)   
 {

}  
 // 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.  
 // Ex) theEnd("Hello", false) -> "o"  
 public static String theEnd(String str, boolean front)  
 {

}  
   
 // Given a string, return true if it ends in "ly".  
 public static boolean endsLy(String str)  
 {

}  
   
 // 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.  
 public static String middleThree(String str)   
 {

}  
   
 // Given two strings, append them together (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".  
 // Ex) conCat("abc", "") -> "abc"  
 public static String conCat(String a, String b)  
 {

}  
 // Given a string, if the string begins with "red" or "blue" return that color string,   
 // otherwise return the empty string.  
 public static String seeColor(String str)  
 {

}  
   
 // Given two strings, append them together (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.  
 public static String minCat(String a, String b)  
 {

}   
   
 // 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.

// Ex) deFront("away") -> "aay"  
 public static String deFront(String str)  
 {

}  
 // 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.   
 // Ex) withoutX2("Hxi") -> "Hi"  
 public static String withoutX2(String str)  
 {

}

// Given a string, return a string where for every char in the original, there are two chars.  
 // Ex) doubleChar("The") -> "TThhee"  
 public static String doubleChar(String str)   
 {

}  
   
 // Return the number of times that the string "hi" appears anywhere in the given string.  
 public static int countHi(String str)   
 {

}  
 // Given two strings, word and a separator sep,   
 // return a big string made of count occurrences of the word,   
 // separated by the separator string.  
 // Ex) repeatSeparator("Word", "X", 3) -> "WordXWordXWord"  
 public static String repeatSeparator(String word,   
 String sep, int count)  
 {

}  
   
 // 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.  
 // Ex) getSandwich("xxbreadjambreadyy") -> "jam"  
 public static String getSandwich(String str)  
 {

}