```
import java.util.Locale;
public class CustomStringMethods
  public CustomStringMethods() { }
  public boolean foundIn(String searchFor, String searchIn) {
       int foundIndex = searchIn.indexOf(searchFor);
       if (foundIndex == -1) {
  public boolean longerThan(String myString, int maxLength) {
```

```
public String funnyString(String str, int idx) {
    String letter = str.substring(idx, idx + 1);
    String returnStr = letter + str + letter;
    return returnStr;
public String pigLatin(String orig)
    String firstLetter = orig.substring(0, 1); // gets the first letter
    String origMinusFirst = orig.substring(1); // gets the string AFTER the first letter
```

// THE 6 NEW METHODS ADDED IN LAB 2 ARE BELOW

```
/**Client provides two strings, strl and str2, and method prints a message to the user that states
whether strl comes before str2, comes after, or they are the same alphabetically.
This method has no return value (void).

Example: if strl is "apple" and str2 is "banana", this method should print a message like:
  "apple comes BEFORE banana alphabetically"

Example: if strl is "banana" and str2 is "apple", this method should print a message like:
  "banana comes AFTER apple alphabetically"

Example: if strl and str2 are both "apple", this method should print a message like:
  "apple and banana are the SAME string!"

*/
```

```
public void alphabetical(String str1, String str2)
   if (str1.equals(str2)) {
       System.out.println(str1 + " and " + str2 + " are the SAME string!");
        int compare = strl.compareTo(str2);
            System.out.println(str1 + " comes BEFORE " + str2 + " alphabetically.");
            System.out.println(str1 + " comes AFTER " + str2 + " alphabetically.");
public String halvesReversed(String myString) {
    int halfIndex = myString.length() / 2;
   String firstHalf = myString.substring(0, halfIndex);
   String secondHalf = myString.substring(halfIndex);
    return secondHalf + firstHalf;
```

```
public String yellOrWhisper(String myString) {
    String firstLetter = myString.substring(0, 1);
    String firstLetterLower = firstLetter.toLowerCase();
    if (firstLetter.equals(firstLetterLower)) {
        return myString.toLowerCase();
        return myString.toUpperCase();
public String endUp(String myString, int numToCap)
    int length = myString.length();
    if (numToCap > length) {
        return myString.toUpperCase();
        String front = myString.substring(0, cut);
        String back = myString.substring(cut);
```

located at removeIdx in myString removed. If removeIdx is outside the bounds of myString, the method should return myString unchanged.

```
public String removeCharacter(String myString, int removeIdx) {
    int maxIndex = myString.length() - 1;
    if (removeIdx > maxIndex) {
        return myString;
        String firstPart = myString.substring(0, removeIdx);
        String secondPart = myString.substring(removeIdx + 1);
        return firstPart + secondPart;
public String insertAt(String orig, String insertText, String searchStr) {
    int foundIndex = orig.indexOf(searchStr);
    if (foundIndex == -1) {
```