Sentence (Recursion) FRQ

This question refers to the `Sentence` class below. Note: A `word` is a string of consecutive nonblank *(and nonwhitespace)* characters. For example, the sentence

"Hello there!" she said.

consists of the four words

"Hello there!" she said.

```
public class Sentence
{
  private String sentence;
  private int numWords;

  /** Constructor. Creates sentence from String str.
   * Finds the number of words in sentence.
   * Preconditlon: Words in str separated by exactly one blank.
   */
  public Sentence(String str)
  { /* to be implemented in part (a) */ }

  public int getNumWords()
  { return numWords; }

  public String getSentence()
  { return sentence; }

  /** Returns a copy of String s with all blanks removed. */
  private static String removeBlanks(String s)
  { /* implementation not shown */ }

  /** Returns a copy of String s with all letters in lowercase.
   * Postcondition: Number of words in returned string equals
   * number of words in s.
   */
  private static String lowerCase(String s)
  { /* implementation not shown */ }

  /** Returns a copy of String s with all punctuation removed.
   * Postcondition: Number of words in returned string equals
   * number of words in s.
   */
  private static String removePunctuation(String s)
  { /* implementation not shown */ }
}
```

(a) Complete the `Sentence` constructor as started below. The constructor assigns `str` to `sentence`. You should write the subsequent code that assigns a value to `numWords`, the number of words in sentence.

Complete the constructor below:

```
/** Constructor. Creates sentence from String str.
 * Finds the number of words in sentence.
 * Precondition: Words in str separated by exactly one blank.
 */
public Sentence(String str){

  sentence = str;
```

Testing Code

Sentence (Recursion) FRQ

(b) Consider the problem of testing whether a string is a palindrome. A palindrome reads the same from left to right and right to left, ignoring spaces, punctuation, and capitalization.

For example,
A Santa lived as a devil at NASA.
Flo, gin is a sin! I golf.
Eva, can I stab bats in a cave?

A public method `isPalindrome` is added to the `Sentence` class. Here is the method and its implementation:

```
/** Returns true if sentence is a palindrome, false otherwise. */
public boolean isPalindrome()
{
  String temp = removeBlanks(sentence);
  Temp = removePunctuation(temp);
  Temp = lowerCase(temp);
  return isPalindrome(temp, 0, temp.length() - 1);
}
```

The overloaded `isPalindrome` method contained in the code is a private recursive helper method, also added to the Sentence class. You are to write the implementation of this method. It takes a "purified" string as a parameter, namely one that has been stripped of blanks and punctuation and is all lowercase letters. It also takes as parameters the first and last index of the string. It returns true if this "purified" string is a palindrome, false otherwise.

A recursive algorithm for testing if a string is a palindrome is as follows:

- If the string has length 0 or 1, it's a palindrome.
- Remove the first and last letters.
- If those two letters are the same, and the remaining string is a palindrome, then the original string is a palindrome. Otherwise it's not.

Complete the `isPalindrome` method below:

```
/** Private recursive helper method that tests whether a substring of
 * string s, starting at start and ending at end, is a palindrome.
 * Returns true if the substring is a palindrome, false otherwise.
 * Precondition: s contains no spaces, punctuation, or capitals.
 */
private static boolean isPalindrome(String s, int start, int end)
```

Testing Code