

1. This question involves the `WordMatch` class, which stores a secret string and provides methods that compare other strings to the secret string. You will write two methods in the `WordMatch` class.

```
public class WordMatch
{
    /** The secret string. */
    private String secret;

    /** Constructs a WordMatch object with the given secret string of lowercase letters. */
    public WordMatch(String word)
    {
        /* implementation not shown */
    }

    /** Returns a score for guess, as described in part (a).
     *   Precondition: 0 < guess.length() <= secret.length()
     */
    public int scoreGuess(String guess)
    { /* to be implemented in part (a) */ }

    /** Returns the better of two guesses, as determined by scoreGuess and the rules for a
     *   tie-breaker that are described in part (b).
     *   Precondition: guess1 and guess2 contain all lowercase letters.
     *                   guess1 is not the same as guess2.
     */
    public String findBetterGuess(String guess1, String guess2)
    { /* to be implemented in part (b) */ }
}
```

**GO ON TO THE NEXT PAGE.**

- (a) Write the `WordMatch` method `scoreGuess`. To determine the score to be returned, `scoreGuess` finds the number of times that `guess` occurs as a substring of `secret` and then multiplies that number by the square of the length of `guess`. Occurrences of `guess` may overlap within `secret`.

Assume that the length of `guess` is less than or equal to the length of `secret` and that `guess` is not an empty string.

The following examples show declarations of a `WordMatch` object. The tables show the outcomes of some possible calls to the `scoreGuess` method.

```
WordMatch game = new WordMatch("mississippi");
```

Value of <code>guess</code>	Number of Substring Occurrences	Score Calculation: (Number of Substring Occurrences) x (Square of the Length of <code>guess</code> )	Return Value of <code>game.scoreGuess(guess)</code>
"i"	4	$4 * 1 * 1 = 4$	4
"iss"	2	$2 * 3 * 3 = 18$	18
"issipp"	1	$1 * 6 * 6 = 36$	36
"mississippi"	1	$1 * 11 * 11 = 121$	121

```
WordMatch game = new WordMatch("aaaabb");
```

Value of <code>guess</code>	Number of Substring Occurrences	Score Calculation: (Number of Substring Occurrences) x (Square of the Length of <code>guess</code> )	Return Value of <code>game.scoreGuess(guess)</code>
"a"	4	$4 * 1 * 1 = 4$	4
"aa"	3	$3 * 2 * 2 = 12$	12
"aaa"	2	$2 * 3 * 3 = 18$	18
"aabb"	1	$1 * 4 * 4 = 16$	16
"c"	0	$0 * 1 * 1 = 0$	0

**GO ON TO THE NEXT PAGE.**

Complete the `scoreGuess` method.

```
/** Returns a score for guess, as described in part (a).  
 * Precondition: 0 < guess.length() <= secret.length()  
 */  
public int scoreGuess(String guess)
```

---

**Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.**

Class information for this question

```
public class WordMatch  
  
private String secret  
  
public WordMatch(String word)  
public int scoreGuess(String guess)  
public String findBetterGuess(String guess1, String guess2)
```

**GO ON TO THE NEXT PAGE.**

- (b) Write the `WordMatch` method `findBetterGuess`, which returns the better guess of its two `String` parameters, `guess1` and `guess2`. If the `scoreGuess` method returns different values for `guess1` and `guess2`, then the guess with the higher score is returned. If the `scoreGuess` method returns the same value for `guess1` and `guess2`, then the alphabetically greater guess is returned.

The following example shows a declaration of a `WordMatch` object and the outcomes of some possible calls to the `scoreGuess` and `findBetterGuess` methods.

```
WordMatch game = new WordMatch("concatenation");
```

Method Call	Return Value	Explanation
<code>game.scoreGuess("ten");</code>	9	1 * 3 * 3
<code>game.scoreGuess("nation");</code>	36	1 * 6 * 6
<code>game.findBetterGuess("ten", "nation");</code>	"nation"	Since <code>scoreGuess</code> returns 36 for "nation" and 9 for "ten", the guess with the greater score, "nation", is returned.
<code>game.scoreGuess("con");</code>	9	1 * 3 * 3
<code>game.scoreGuess("cat");</code>	9	1 * 3 * 3
<code>game.findBetterGuess("con", "cat");</code>	"con"	Since <code>scoreGuess</code> returns 9 for both "con" and "cat", the alphabetically greater guess, "con", is returned.

**GO ON TO THE NEXT PAGE.**

Complete method `findBetterGuess`.

Assume that `scoreGuess` works as specified, regardless of what you wrote in part (a). You must use `scoreGuess` appropriately to receive full credit.

```
/** Returns the better of two guesses, as determined by scoreGuess and the rules for a
 * tie-breaker that are described in part (b).
 * Precondition: guess1 and guess2 contain all lowercase letters.
 * guess1 is not the same as guess2.
 */
public String findBetterGuess(String guess1, String guess2)
```

---

**Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.**

Class information for this question

```
public class WordMatch

private String secret

public WordMatch(String word)
public int scoreGuess(String guess)
public String findBetterGuess(String guess1, String guess2)
```

**GO ON TO THE NEXT PAGE.**