Free-Response Question

Free-response questions related to this lab will typically focus on text processing. One possible question is shown below.

When Web developers want to convert plain text documents to HTML, they need to make changes to the text to insert tags and replace certain characters. For this question, you will write part of the class to convert text to HTML. You will write methods to replace underscores (`"_"`) with tags indicating the text should be in italics.

The class definition for this problem is given below:

```
public class TextFormatter
{
  private String line; //  The line to format

  public TextFormatter (String lineToFormat)
  {  line = lineToFormat;  }


  /**
   * Finds the first single instance of str in line,
   * starting at the position start
   * @param str the string of length 1 to find.
   * Guaranteed to be length 1.
   * @param start the position to start searching.
   * Guaranteed to be in the string line.
   * @return the index of the first single instance of
   * str if the string is found or -1 if it is not found.
   */
  private int findString (String str, int start)
  {  /* To be implemented in part a) */ }


  /**
   * Count the number of times single instances of str appear in
   * the line.
   * @param str the string to find.
   * Guaranteed to be length 1.
   * @return the number of times the string appears in the line
   */
  private int countStrings (String str)
  {  /* To be implemented in part b) */ }


  /**
   * Replace all single instances of underscores in the line given by
   * line with italics tags.  There must be an even number of underscores
```

```
   * in the line, and they will be replaced by <I>, </I>, alternating.
   * @param original a string of length 1 to replace
   * @param replacement the string (of any length) use as a replacement
   * @return the line with single instances of underscores replaced with
   * <I> tags or the original line if there are not an even number of
   * underscores.
   */
  public String convertItalics ()
  {  /* To be implemented in part c) */ }
}
```

a) Write the method `findString`. This method will take a string of length 1 to find in the line, at a given starting point. It will return the location of the goal string. This differs from the `indexOf` method of the `String` class because it requires the string be just a single instance of the string, so if the string appears two or more times consecutively, it will not return any of those values. If there is no single instance, the method should return -1. Consider the following examples, where `line` has the value `"aabaccb"`.

Call	Result	Explanation
`findString("a", 0)`	3	The first single occurrence of `"a"` is in position 3. The `"a"`s in position 0 and 1 are not returned because they are not single instances.
`findString("b", 4)`	6	The first single occurrence of `"b"` at or after position 4 is in position 6.
`findString("c", 0)`	-1	There is no single instance of `"c"` in the line.

b) Write the method `countStrings`. This method will take a string of length 1 to find in the line. It will return the number of times single instances of the goal string appear in the string.

c) Write the method `convertItalics`. If the line has an even number of single underscores, a line with those underscores converted to alternating <I> and </I> tags should be returned. The first underscore will be converted to <I>, the second to </I>, the third to <I>, and so on. If the line does not have an even number of <I> tags, the original line should be returned. Consider the following examples.

line	value returned by `convertItalics`
`This is _very_ good.`	`This is <I>very</I> good.`
`_This_ is _very_ _good_.`	`<I>This</I> is <I>very</I> <I>good</I>.`
`This is _very good.`	`This is _very good.`
`This is __very good.`	`This is __very good.`

Possible Solution to the Free-Response Question

The complete `TextFormatter` class is below. Teachers who want to go further with this example, as a homework assignment or class example, can add handling asterisks for the <B> tag and/or substituting for special characters (for example, `"<"` is used instead of a less than sign—`"<"`). While just adding a second tag like an asterisk seems to be simple, extra care will need to be taken to ensure that tags are nested correctly. For example, the line `"Here *is a _bad* example_"` has invalid nesting.

```
public class TextFormatter
{
  private String line; //  The line to format

  public TextFormatter (String lineToFormat)
  {  line = lineToFormat;  }


  /**
   * Finds the first single instance of str in line,
   * starting at the position start
   * @param str the string of length 1 to find.
   * Guaranteed to be length 1.
   * @param start the position to start searching.
   * Guaranteed to be in the string line.
   * @return the index of the first single instance of
   * str if the string is found or -1 if it is not found.
   */
  private int findString (String str, int start)
  {
    int psn = line.indexOf(str, start);
    while (psn >= 0) {
      //  find the string of length 1 before and after the instance found
      String before = "";
      String after = "";
      if (psn > 0)
        before = line.substring(psn - 1, psn);
```

```
    if (psn < line.length() - 1)
      after = line.substring(psn + 1, psn + 2);

    //  If the before and after are different from
    //  this string, we've found the position.
    if (!before.equals(str) && !after.equals(str))
      return psn;

    //  If not, look for the next position
    psn = line.indexOf (str, psn + 1);
  }
  //  Otherwise, the string isn't found.
  return -1;
}

/**
 * Count the number of times single instances of str appear in
 * the line.
 * @param str the string to find.
 * Guaranteed to be length 1.
 * @return the number of times the string appears in the line
 */
private int countStrings (String str)
{
  int count = 0;
  int psn = 0;
  while (findString (str, psn) >= 0)
  {
    count++;
    psn = findString (str, psn) +1;
  }
  return count;
}

/**
 * Replace all single instances of underscores in the line given by
 * line with italics tags.  There must be an even number of underscores
 * in the line, and they will be replaced by <I>, </I>, alternating.
 * @param original a string of length 1 to replace
 * @param replacement the string (of any length) use as a replacement
 * @return the line with single instances of underscores replaced with
 * <I> tags or the original line if there are not an even number of
 * underscores.
 */
public String convertItalics ()
{
  //  Ensure there are an even number of underscores.
  if (countStrings ("_") % 2 == 1)
    return line;
```

```
    String tag = "<I>";
    String result = "";
    int psn = 0;

    while (findString ("_", psn) >= 0 )
    {
      int newPsn = findString ("_", psn);
      //  Copy the code before the "_" into the result
      result = result + line.substring(psn, newPsn);

      //  Add the tag and change the tag
      result = result + tag;
      if (tag.equals ("<I>"))
        tag = "</I>";
      else
        tag = "<I>";

      //  update the psn
      psn = newPsn + 1;
    }
    //  copy the rest of the string
    result = result + line.substring(psn);

    return result;
  }

}
```