3. This question involves analyzing and modifying a string. The following `Phrase` class maintains a phrase in an instance variable and has methods that access and make changes to the phrase. You will write two methods of the `Phrase` class.

```
public class Phrase
{
   private String currentPhrase;

   /** Constructs a new Phrase object. */
   public Phrase(String p)
   {  currentPhrase = p;  }


   /** Returns the index of the nth occurrence of str in the current phrase;
    *  returns -1 if the nth occurrence does not exist.
    *  Precondition: str.length() > 0 and n > 0
    *  Postcondition: the current phrase is not modified.
    */
   public int findNthOccurrence(String str, int n)
   {  /* implementation not shown */  }


   /** Modifies the current phrase by replacing the nth occurrence of str with repl.
    *  If the nth occurrence does not exist, the current phrase is unchanged.
    *  Precondition: str.length() > 0 and n > 0
    */
   public void replaceNthOccurrence(String str, int n, String repl)
   {  /* to be implemented in part (a) */  }


   /** Returns the index of the last occurrence of str in the current phrase;
    *  returns -1 if str is not found.
    *  Precondition: str.length() > 0
    *  Postcondition: the current phrase is not modified.
    */
   public int findLastOccurrence(String str)
   {  /* to be implemented in part (b) */  }


   /** Returns a string containing the current phrase. */
   public String toString()
   {  return currentPhrase;  }
}
```

Part (a) begins on page 12.

(a) Write the `Phrase` method `replaceNthOccurrence`, which will replace the `nth` occurrence of the string `str` with the string `repl`. If the `nth` occurrence does not exist, `currentPhrase` remains unchanged.

Several examples of the behavior of the method `replaceNthOccurrence` are shown below.

Code segments	Output produced

`Phrase phrase1 = new Phrase("A cat ate late.");`
`phrase1.replaceNthOccurrence("at", 1, "rane");`
`System.out.println(phrase1);`
	`A crane ate late.`

`Phrase phrase2 = new Phrase("A cat ate late.");`
`phrase2.replaceNthOccurrence("at", 6, "xx");`
`System.out.println(phrase2);`
	`A cat ate late.`

`Phrase phrase3 = new Phrase("A cat ate late.");`
`phrase3.replaceNthOccurrence("bat", 2, "xx");`
`System.out.println(phrase3);`
	`A cat ate late.`

`Phrase phrase4 = new Phrase("aaaa");`
`phrase4.replaceNthOccurrence("aa", 1, "xx");`
`System.out.println(phrase4);`
	`xxaa`

`Phrase phrase5 = new Phrase("aaaa");`
`phrase5.replaceNthOccurrence("aa", 2, "bbb");`
`System.out.println(phrase5);`
	`abbba`

Class information for this question

```
public class Phrase

private String currentPhrase
public Phrase(String p)
public int findNthOccurrence(String str, int n)
public void replaceNthOccurrence(String str, int n, String repl)
public int findLastOccurrence(String str)
public String toString()
```

The `Phrase` class includes the method `findNthOccurrence`, which returns the `nth` occurrence of a given string. You must use `findNthOccurrence` appropriately to receive full credit.

Complete method `replaceNthOccurrence` below.

```
/** Modifies the current phrase by replacing the nth occurrence of str with repl.
 *  If the nth occurrence does not exist, the current phrase is unchanged.
 *  Precondition: str.length() > 0 and n > 0
 */
public void replaceNthOccurrence(String str, int n, String repl)
```

Part (b) begins on page 14.

AP® COMPUTER SCIENCE A
2017 SCORING GUIDELINES

Question 3: PhraseEditor

Part (a)	`replaceNthOccurrence`	**5 points**

Intent: *Replace the nth occurrence of a given string with a given replacement*

+1 Calls `findNthOccurrence` to find the index of the `nth` occurrence

+1 Preserves `currentPhrase` only if `nth` occurrence does not exist

+1 Identifies components of `currentPhrase` to retain (uses `substring` to extract before/after)

+1 Creates replacement string using identified components and `repl`

+1 Assigns replacement string to instance variable (`currentPhrase`)

Part (b)	`findLastOccurrence`	**4 points**

Intent: *Return the index of the last occurrence of a given string*

+1 Calls `findNthOccurrence` to find the index of the *n*th occurrence

+1 Increments (or decrements) the value used as *n* when finding *n*th occurrence

+1 Returns the index of the last occurrence, if it exists

+1 Returns -1 only when no occurrences exist

Question-Specific Penalties

-1 (q) Uses `currentPhrase.findNthOccurrence`

-2 (r) Confused identifier instead of `currentPhrase`

AP® COMPUTER SCIENCE A
2017 SCORING GUIDELINES

Question 3: Scoring Notes

Part (a) `replaceNthOccurrence`			**5 points**
Points	Rubric Criteria	Responses earn the point if they ...	Responses will not earn the point if they ...
+1	Calls `findNthOccurrence` to find the index of the `nth` occurrence	• do not use the result of calling `findNthOccurrence`	
+1	Preserves `currentPhrase` only if `nth` occurrence does not exist		• fail to use a conditional
+1	Identifies components of `currentPhrase` to retain (uses `substring` to extract before/after)	• identify start and end of substring to be replaced	
+1	Creates replacement string using identified components and `repl`		• create a replacement string that is out of order
+1	Assigns replacement string to instance variable (`currentPhrase`)		
Part (b) `findLastOccurrence`			**4 points**
Points	Rubric Criteria	Responses earn the point if they ...	Responses will not earn the point if they ...
+1	Calls `findNthOccurrence` to find the index of the *n*th occurrence	• do not use the result of calling `findNthOccurrence`	• `return currentPhrase.lastIndexOf(str);` • call `findNthOccurrence` with an integer parameter of 0
+1	Increments (or decrements) the value used as *n* when finding *n*th occurrence	• `return currentPhrase.lastIndexOf(str);` • advance through `currentPhrase` searching for *n*th occurrence of `str`	
+1	Returns the index of the last occurrence, if it exists	• `return currentPhrase.lastIndexOf(str);` • compute the correct value to be returned in all cases, but no return statement exists for any case	• shorten string being searched • always return in first iteration of the loop
+1	Returns -1 only when no occurrences exist	• `return currentPhrase.lastIndexOf(str);`	• compute the correct value to be returned in all cases, but no return statement exists for any case • always return in first iteration of the loop