1. What is the output of the following code segment:

```
String s = "ban";
ArrayList<String> words = new ArrayList<String>();
words.add(s);
words.add(s.substring(1));
words.add(s.substring(1,2));
String w = "";
for(int k=0; k<words.size(); k++)
{
   w+= words.get(k);
}
System.out.println(w.indexOf("an"));
```

(a) 1 (b) 2 (c) 3 (d) ana (e) banana

2. What is the output of the following code segment:

```
ArrayList<String> list = new ArrayList<String>();
list.add("A");
list.add("B");
list.add("C");
list.add("D");
list.add("E");

for(int k=1; k<=3; k++)
{
   list.remove(1);
}

for(int k=1; k<=3; k++)
{
   list.add(1, "*");
}

for(int k=0; k<list.size(); k++)
{
   System.out.println(list.get(k) + " ");
}
```

(a) `A C D E * * *`
(b) `* * * B C D E`
(c) `A * * * E`
(d) `A E * * *`
(e) `IndexOutOfBoundsException`

3. Consider the following code segment with a missing "for" loop:

```
ArrayList<String> letters = new ArrayList<String>();
letters.add("A");
letters.add("B");
letters.add("C");

<missing "for" loop>

System.out.println(letters);
```

Suppose, when executed, the above code segment displays:

```
[A*, B*, C*]
```

Which of the following could replace `<missing "for" loop>`?

I.	`for(int i=0; i<letters.size(); i++)`
	`{`
	`  letters.set(i, letters.get(i) + "*");`
	`}`

II.	`for(int i=0; i<letters.size(); i++)`
	`{`
	`  String s = letters.get(i);`
	`  s = s + "*";`
	`}`

III.	`for(int i=0; i<letters.size(); i++)`
	`{`
	`  String s = letters.remove(i);`
	`  s = s + "*";`
	`  letters.add(i, s)`
	`}`

(a) I only
(b) II only
(c) I and II only
(d) I and III only
(e) II and III only

Write the body for each method. Declare all necessary local variables.

1. Write a void method *replaceAll* that replaces all occurrences of a given word in an ArrayList with another word. If the first word does not exist in the ArrayList, then nothing should happen. For example, if a call is made to *replaceAll* `(list, "pen", "pencil")`, the word "pencil" will replace all occurrences of the word "pen" in the list, if the word "pen" is in the list.

   Use the method header given below:

   ```
   // Pre-Condition: list is a non-empty list of String objects
   // Post-Condition: list is modified by replacing word1 with word2
   public static void replaceAll(ArrayList list, String word1, String word2)
   ```

2. Write a void method *insertWords* that inserts a given *String* at the front of an ArrayList until the list has a given size. For example, if the list originally consists of {"book", "pencil", "eraser"} and a call of `insertWords(list, "pen", 8)` is made, the list will then be modified to

   {"pen", "pen", "pen", "pen", "pen", "book", "pencil", "eraser"}

   So the list now has 8 words in it.

   Note: If the size of the list is already greater than or equal to *newSize*, then no words should be inserted in the list.

   Use the method header given below:

   ```
   // Pre-Condition: list is a non-empty list of String objects
   // Post -Condition: if the size of list is less than newSize, the list is modified by inserting
   // the appropriate number of Strings word until the size of the ArrayList equals newSize
   // the words are to be inserted at the front of the ArrayList
   public static void insertWords(ArrayList list, String word, int newSize)
   ```

3. Write a void method *insertWord* that inserts a word in front of another word in an ArrayList. If the word does not exist in the ArrayList, then nothing should happen. It is also to be assumed that the word will occur at most once in the list. For example, if a call is made to `insertWord(list,"pen","pencil")`, the word "pencil" will be inserted in front of the word "pen" in the list, if the word "pen" is in the list.

   Use the method header given below:

   ```
   // Pre-Condition: list is a non-empty list of String objects
   // Post -Condition: list is modified by inserting word2 in front of word1
   public static void insertWord(ArrayList list, String word1, String word2)
   ```