In this exercise you were given four methods to define, focusing on transforming values passed in to the methods through parameters to produce return values, often using for loops. There were two versions of this exercise – though the scenario was a little different in both cases the structure of the solution was the same in both.

METHOD 1

The first method was described as follows:

```
/* 75 COMPETENCY POINTS
* Question 1: Define this method so it returns those strings from the
* ArrayList<String> named list that contain the String named s.  There
* are (at least) two ways to do this:
*
* 1) use the contains(String) -> boolean method
*
*       Ex. "Bookkeeper".contains("keep") returns true
*       Ex. "Bookkeeper".contains("jdgfgj") returns false
*
* 2) use the indexOf(String) -> int method
*
*    Ex. "Bookkeeper".indexOf("keep") returns 4
*    Ex. "Bookkeeper".indexOf("jdgfgj") returns -1
*
* Example 1: If list is ["Leela", "Fry", "Zoidberg", "Bender"] and s is "er"
* then the answer is ["Zoidberg", "Bender"].
*
* Example 2: If list is ["Leela", "Fry", "Zoidberg", "Bender"] and s is "e"
* then the answer is ["Leela", "Zoidberg", "Bender"].
*
* Example 3: If list is ["Leela", "Fry", "Zoidberg", "Bender"] and s is "xyz"
* then the answer is [].
*
*/
```

Here's the method stub you were provided with:

```
public ArrayList<String> question1(ArrayList<String> list, String s) {
       ArrayList<String> answer = new ArrayList<String>();
       return answer;  // edit this method to return the correct value in answer
}
```

The basic function of this method is to ***filter*** the input ArrayList to produce another ArrayList consisting of a subset of the values in the original. This is a pattern that we will encounter repeatedly (see the HW2 retrospective, for example).

The basic approach to writing a filtering method is to write a loop to process each member of the input ArrayList, and to incorporate an if statement that determines whether or not to add a given item to the answer ArrayList.

The provided "stub" creates the ArrayList in which we will accumulate the answer:

```
public ArrayList<String> question1(ArrayList<String> list, String s) {
	ArrayList<String> answer = new ArrayList<String>();









	return answer;
}
```

We add a for loop to process each item of the input `list`:

```
public ArrayList<String> question1(ArrayList<String> list, String s) {
	ArrayList<String> answer = new ArrayList<String>();
	for (int i=0; i<list.size(); i=i+1) {
		String t = list.get(i);



	}
	return answer;
}
```

Finally, we add a conditional statement that determines whether the String t should be added to the ArrayList answer:

```
public ArrayList<String> question1(ArrayList<String> list, String s) {
	ArrayList<String> answer = new ArrayList<String>();
	for (int i=0; i<list.size(); i=i+1) {
		String t = list.get(i);
		if (t.contains(s)) {
			answer.add(t);
		}
	}
	return answer;
}
```

METHOD 2

The second method was given to you:

```
/* 75 COMPETENCY POINTS
* Question 2:
*
* Compute the sum of the lengths of the Strings at every other index,
* starting at index start.
*
* If there are no Strings at those indices return -1.
*
* Example 1: If list is ["Leela", "Fry", "Zoidberg", "Bender"] and start is 1
* then the answer is 9, because "Fry" is at index 1 and has length 3,
* while "Bender" is at index 3 and has length 6.
*
* Example 2: If list is ["Leela", "", "Zoidberg"] and start is 1
* then the answer is 0, because "" is at index 1 and has length 0.
*
* Example 3: If list is ["Leela", "", "Zoidberg"] and start is 2
* then the answer is 8, because "Zoidberg" is at index 2 and has length 8.
*
* Example 4: If list is ["Leela"] and start is 2
* then the answer is -1, because there are no Strings at or after index 2.
*
*/
```

The method stub just returns zero:

```
public int question2(ArrayList<String> list, int start) {
	return 0;  // edit this method to return the correct value
}
```

The method needs to return an int value, so let's declare a variable to hold that value:

```
public int question2(ArrayList<String> list, int start) {
	int answer = 0;









	return answer;
}
```

If the loop would not be entered, return -1:

```
public int question2(ArrayList<String> list, int start) {
	int answer = 0;
	if (! (start < list.size())) {
		return -1;
	}
```

```
        return answer;
}
```

Write the loop to inspect each String in the ArrayList:

```
public int question2(ArrayList<String> list, int start) {
        int answer = 0;
        if (! (start < list.size())) {
                return -1;
        }
        for (int i=start; i<list.size(); i=i+2) {
                String s = list.get(i);

        }
        return answer;
}
```

Finally, accumulate the lengths of the selected Strings in answer:

```
public int question2(ArrayList<String> list, int start) {
        int answer = 0;
        if (! (start < list.size())) {
                return -1;
        }
        for (int i=start; i<list.size(); i=i+2) {
                String s = list.get(i);
                answer = answer +s.length();
        }
        return answer;
}
```

METHOD 3

The third method was described as follows:

```
/* 2 PROFICIENCY POINTS
* Question 3:
*
* If the list has a String that contains the String s, then return
* the highest index of such a String.
* If the list does not have such a String, return -1.
*
* Example 1: If list is ["Leela", "Fry", "Zoidberg", "Bender"] and s is "er"
* then the answer is 3.
*
* Example 2: If list is ["Leela", "Fry", "Zoidberg", "Bender"] and s is "y"
* then the answer is 1.
*
* Example 3: If list is ["Leela", "Fry", "Zoidberg", "Bender"] and s is "xyz"
* then the answer is -1.
```

```
	*/
```

The given stub again simply returns zero:

```
	public int question3(ArrayList<String> list, String s) {
		return 0;  // edit this method to return the correct value
	}
```

One approach to this method is to loop through the Strings in the ArrayList starting at high indices and moving towards low indices, rather than the other way around:

```
	public int question3(ArrayList<String> list, String s) {
		for (int i=list.size()-1; i>=0; i=i-1) {
			String t = list.get(i);



		}

	}
```

If we find a String t that contains s we can immediately return the index where t was found – this will be the highest index:

```
	public int question3(ArrayList<String> list, String s) {
		for (int i=list.size()-1; i>=0; i=i-1) {
			String t = list.get(i);
			if (t.contains(s)) {
				return i;
			}
		}

	}
```

If the loop exits normally that means that no String containing s was found anywhere in the ArrayList, so we return -1:

```
	public int question3(ArrayList<String> list, String s) {
		for (int i=list.size()-1; i>=0; i=i-1) {
			String t = list.get(i);
			if (t.contains(s)) {
				return i;
			}
		}
		return -1;
	}
```